UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FINAL DE MÁSTER

Desarrollo y validación de módulos en la arquitectura ABNO

Autor: Marta Cuaresma Saturio

Tutor: Víctor López Álvarez

Ponente: Jorge E. López de Vergara Méndez

Octubre 2013

RESUMEN

En este trabajo final de máster se ha contribuido a dos módulos de la arquitectura Application-Based Network Operations (ABNO): el Path Computation Element (PCE) y el módulo de topología, el cuál además se ha implementado por completo. ABNO engloba distintas tecnologías que recogen la información sobre los recursos disponibles en la red con el objetivo de proporcionar rutas para el tráfico. ABNO puede verse como una colaboración entre una serie de elementos ya existentes y nuevos elementos propios de la arquitectura. El componente principal es el PCE, que es el encargado del cálculo de las rutas permitiendo ejecutar además operaciones de ingeniería de tráfico. Éste se coordina con los demás elementos de la arquitectura, consiguiendo así, automatizar el proceso de configuración de los caminos en la red.

El primer módulo al que se ha contribuido es el PCE, en el cual se ha estudiado el problema conocido como "robo de lambda". El PCE almacena la topología de red en la Traffic Engineering Databse (TED), que puede actualizarse de distintas formas. Este proceso de actualización lleva un tiempo, dando lugar a que la TED pueda estar desactualizada con respecto al estado real de la red. Esto provoca el bloqueo por robo de lambda que se puede evitar con el mecanismo de pre-reserva.

Este mecanismo de pre-reserva se ha evaluado en dos casos de uso: restauración masiva y múltiples PCEs. El primero de ellos estudia qué ventajas ofrece este mecanismo en un escenario de restauración masiva de la red. Se observará que este mecanismo reduce considerablemente el número de reintentos de peticiones y el tiempo de restablecimiento del tráfico. El segundo escenario es el caso de múltiples PCEs calculando rutas dentro de un mismo dominio de red. Se han estudiado varios escenarios posibles: PCEs independientes, PCEs encargados de recursos independientes de la red y PCEs colaborativos. Para este último se ha extendido el protocolo PCEP para permitir la comunicación y envío de mensajes de notificación entre los PCEs colaborativos.

El segundo módulo dentro de la arquitectura ABNO al que se ha contribuido es el módulo de topología que se ha diseñado, implementado y evaluado. Éste módulo es el encargado de almacenar la topología de la red, mantenerla actualizada y así poder proporcionar dicha información a otros módulos de la arquitectura. La topología puede ser actualizada de distintas formas, mediante la lectura de un fichero XML, por *web service* o procesando los mensajes de enrutamiento como OSPF o BGP-LS. El módulo de topología se ha evaluado en dos escenarios multicapa y multidominio.

PALABRAS CLAVE

ABNO, PCE, módulo de topología, cálculo de caminos, BGP-LS, bloqueo por robo de lambda, probabilidad de bloqueo, multidominio, multicapa.

ABSTRACT

This Master Thesis contributes with two modules belonging to Application-Based Network Operations (ABNO) architecture: Path Computation Element (PCE) and Topology Module. The last one has been fully implemented. ABNO gathers different technologies together to store the topology network information and the available resources to provide paths to accommodate the traffic in the network. The main component is the PCE which is in charge of calculating paths in the network and it can apply some restrictions and policies in the computation. It coordinates with the others components achieving an automated configuration process for managing the network.

The first contribution is the PCE where it has been studied the "stolen lambda" problem. The PCE stores the network topology in the Traffic Engineering Database (TED), which can be updated from different ways. This updating process takes some time, which causes the network status information no to be fully consistent with the TED information. Therefore, the PCE can compute paths using resources already used in the network which were not updated in time. This causes a stolen-lambda block. The Pre-Reservation mechanism has been proposed to eliminate this block.

The Pre-Reservation mechanism is evaluated in two uses cases, a massive restoration scenario and multiple PCEs scenario. The former studies the advantages of this mechanism in a massive restoration scenario and it will be prove that the stolen-lambda block decreases significantly and therefore the time taking to restore the traffics. The other scenario studies the performance achieved by multiple PCEs, which implement the Pre-Reservation mechanism, computing routes in the same domain. Three possible scenarios have been defined and evaluated: independent PCEs, lambda-subset PCEs and collaborative PCEs. The PCEP protocol has been extended to allow communication between the collaborative PCEs.

Finally, a topology module for the ABNO architecture has been implemented, tested and evaluated. This module stores the topology network information which can be updated reading a XML file, by *web service* or processing routing messages from the network elements, like OSPF or BGP-LS. Two scenarios have been studied, a multi-layer and a multi-domain scenario.

KEY WORDS

ABNO, PCE, topology module, path computation, BGP-LS, stolen-lambda block, blocking probability, multi-domain, multi-layer.

AGRADECIMIENTOS

En primer lugar quiero agradecer a mi familia por el esfuerzo que han puesto en darme la mejor educación y apoyarme siempre en todo lo que he hecho. Gracias a mis padres por darme la posibilidad de estudiar la carrera y el máster que quise, e incluso, irme un año a estudiar fuera de España. Gracias a ellos por ayudarme en todo momento. Gracias a mi hermana por todo el cariño que me ha dado siempre. Por ser tan buena estudiante y tener tanta paciencia en mis momentos de estrés.

En segundo lugar quiero agradecer a mis amigos de la universidad, María, Susana, Paula, Gonzalo, Carlos G., Pablo y Carlos R. por hacerme mi vida universitaria más fácil y divertida. Por ayudarme y ser grandes compañeros de prácticas. Han sido 5 años conviviendo día a día que recordaré siempre.

Quería agradecer también a mi tutor, Víctor López, por darme la posibilidad de realizar este proyecto, enseñarme y apoyarme durante la consecución del mismo. Gracias a mis compañeros de trabajo, y en especial a mis compañeros Sergio y Arturo. Ha sido un placer poder trabajar con todos vosotros.

Por último, agradecer a Jorge E. López por ser un gran tutor y profesor tanto en la carrera como en el máster, haberme ayudado con el proyecto y haber resuelto con presteza todas mis dudas.

A todos, gracias.

ÍNDICE

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ACRÓNIMOS

ABNO	Application-Based Network Operations
AS	Sistema Autónomo
BGP	Border Gateway Protocol
BGP-LS	Border Gateway Protocol Link State
DB	Database
DMDMTD	Distributed Multi-Domain Minimum Transit Domains
ERO	Explicit Route Object
GMPLS	Generalized Multiprotocol Label Switching
H-PCE	Hierarchical PCE
IGP	Interior Gateway Protocol
KSP	K-Shortest-Path
LMDMTD	Local Multi-Domain Minimum Transit
LS	Link State
LSA	Link State Advertisement
LSP	Label Switched Path
LSR	Labeled Switch Router
MPLS	Multiprotocol Label Switching
MTBF	Mean Time Between Failures
MTTR	Mean Time To Repair
NE	Network Emulator
NLRI	Network Layer Reachability Information
NMS	Network Management System
OAM	Operaciones, Administración y Mantenimiento
OSPF	Open Shortest Path First
OSS	Operations Support System
PCC	Path Computation Client
PCE	Path Computation Element
PCEP	Path Computation Element Communication Protocol
PR	Pre-Reserva
RFC	Request for Comments
RP	Request Parameters
RSVP	Resource ReSerVation Protocol
RTT	Round-Trip delay time

SP	Shortest-Path
TCP	Transmission Control Protocol
TDM	Time-Division Multiplexing
TE	Traffic Engineering
TED	Traffic Engineering Database
TID	Telefónica I+D
VNTM	Virtual Network Topology Manager
VPN	Virtual Private Network
WDM	Wavelength-Division Multiplexing
WSON	Wavelength Switched Optical Networks

1. Introducción

1.1. Motivación

Las redes de hoy en día integran múltiples tecnologías que permiten ofrecer una variedad de servicios adaptados a las características y demandas de las distintas aplicaciones. Hay un incremento en la demanda de tráfico debido a los nuevos servicios que ofrece Internet, los cuales necesitan bajo demanda conectividad, fiabilidad y recursos en un rango de tecnologías de red desde la capa IP/MPLS hasta la capa óptica. Es por ello que durante los últimos años se ha hecho un gran esfuerzo en la definición del plano de control para redes de transporte. Sin embargo, cuando existen varios fabricantes y tecnologías la interoperabilidad entre los distintos dominios o capas se hace complicada. Un punto importante en el que centrarse es cómo gestionar el plano de control. Al haber tanta heterogeneidad en tecnologías, definir una automatización del plano de control es esencial para evitar posibles problemas y reducir los costes de operación de la red. Recientemente, se ha propuesto en el IETF la arquitectura Application-Based Network Operations (ABNO) [1] que permite realizar de forma automática operaciones que hasta ahora requieren intervención manual o no existían interfaces estándares para realizarlo.

ABNO puede proporcionar los siguientes tipos de servicio a las aplicaciones:

- Optimización de los flujos de tráfico entre aplicaciones.
- Control remoto de los componentes de la red.
- Interconexión CDN.
- Coordinación de los recursos de la red.
- Redes virtuales privadas (VPN).

En este trabajo final de máster se contribuirá a dos módulos de la arquitectura ABNO: un Path Computation Element (PCE) y un módulo de Bases de Datos.

El PCE es una componente fundamental de la arquitectura ABNO y fue definida por el IETF [2]. El PCE es un componente encargado del cálculo de rutas basándose en el estado de la red y que puede aplicar en sus cálculos ciertas restricciones. Fue desarrollado para el cálculo de caminos en redes MPLS y GMPLS.

La arquitectura ABNO incluye una serie de bases de datos que contienen la información necesaria para el sistema. Las dos principales son la Traffic Engineering Database (TED) y la Label Switched Path Database (LSP-DB). La TED almacena la información de los recursos de la red indicando cuáles están disponibles y cuáles no y puede contener información sobre métricas, capacidad de ancho de banda de los enlaces, etc. La TED puede construirse o actualizarse por medio de peticiones directas a través del Network Management System (NMS) o a través de mensajes de enrutamiento leídos directamente desde la red. En el proyecto se implementará este módulo en su totalidad que será llamado Módulo de Topología (Topology Module).

La LSP-DB guarda información sobre los LSPs que están activos en la red o que podrán establecerse. Esta base de datos no será implementada en el proyecto, ya que se usa solo cuando hay un PCE activo y en el sistema actual no existe tal PCE.

Desde el punto de vista funcional, existe un Path Computation Client (PCC) que pide nuevas rutas al PCE con las restricciones necesarias. El protocolo que permite la comunicación entre PCE y PCC es el Path Computation Element Protocol (PCEP) [3]. El PCE computa las rutas basándose en su TED que se actualiza con la información enviada por los protocolos de enrutamiento. Esta actualización no es instantánea, lo que ocasiona que la TED pueda estar desactualizada con respecto a la información del plano de control. Esto ocasiona que el PCE compute un camino óptico con recursos de red ocupados que él consideraba libres. Este problema se denomina "robo de lambda" [4] y puede ser eliminado añadiendo un mecanismo de pre-reserva de recursos (PR), propuesto en [5]. En este trabajo final de master se han

realizado mejoras para solucionar este problema del "robo de lambda" en entornos de múltiples PCEs y en caso de restauración masiva.

1.2. Objetivos

La finalidad de este trabajo de fin de máster es el desarrollo, implementación y validación de dos módulos para la arquitectura ABNO, el PCE y el Módulo de Topología.

Los objetivos principales son:

- Diseño y validación del módulo PCE de la arquitectura ABNO. Se utilizará el PCE implementado en Telefónica I+D (TID). Éste PCE implementará el mecanismo de pre-reserva. Se evaluarán dos casos de uso:
    - Evaluación del mecanismo de pre-reserva en entornos de restauración masiva.
        - Se implementará un PCC encargado de hacer las peticiones del enlace caído.
        - Se medirán y evaluarán los resultados obtenidos mediante emulación, variando el tiempo de pre-reserva y variando el tiempo de configuración de los ROADMs.
        - Se compararán los resultados con aquellos obtenidos sin el mecanismo de pre-reserva.
    - Evaluación del mecanismo de pre-reserva en entornos de múltiples PCEs
        - Se estudiarán los posibles entornos de múltiples PCEs.
        - Se implementará un PCC encargado de hacer las peticiones a los distintos PCEs. El PCC podrá implementar balanceo de carga.
        - Se definirán unas extensiones al protocolo PCEP para poder desarrollar el entorno de PCEs colaborativos.
- Diseño, implementación y validación del Módulo de Topología de la arquitectura ABNO. Éste módulo será el encargado de almacenar la topología de la red y de proporcionar su información a otros módulos de la arquitectura. La topología podrá crearse y actualizarse mediante peticiones a través de *web service*, leyendo un fichero XML, o por mensajes de enrutamiento, como el OSPF o el BGP-LS.

1.3. Estructura de esta memoria

La memoria se estructura como sigue, en la sección 2 se introduce el modelo de arquitectura ABNO, sus componentes y se explica en detalle la arquitectura del PCE. En la sección 3 se explica en detalle el mecanismo de pre-reserva y se evalúa en un escenario de restauración masiva de la red, donde se emulará un fallo de un enlace y la restauración de todo el tráfico que ocupaba ese enlace. Éste mecanismo también se evalúa en un escenario con múltiples PCEs en un dominio, sección 4. En la sección 5 se expone el diseño y la implementación de un Módulo de Topología para la arquitectura ABNO. Finalmente, se concluye la memoria y se muestra el posible trabajo futuro, sección 6.

2. Modelo de arquitectura ABNO

El Aplication-Based Network Operator (ABNO) es una arquitectura de red compuesta por varios módulos y tecnologías que reúnen información acerca de la topología y los recursos disponibles en la red, con el objetivo de proporcionar rutas para el tráfico y de reservar los recursos necesarios para ellas. ABNO puede verse como una colaboración entre una serie de elementos ya existentes y nuevos elementos propios de la arquitectura que se coordinan para gestionar las conexiones dentro de una red de una manera óptima y automática. El componente clave en la arquitectura ABNO es el Path Computation Element (PCE), que es el encargado de computar los caminos y es además extendido para proveer políticas y restricciones en el cálculo de ellos.

En este apartado se describe la arquitectura ABNO, sus componentes y la conexión entre ellos. Además, se explicará más en detalle la arquitectura del PCE.

2.1. Motivación

La definición de esta arquitectura viene motivada por el crecimiento de las aplicaciones y servicios de internet que necesitan, bajo demanda, conectividad de red, reserva de recursos, etc. Además, estos recursos pueden ser de múltiples tecnologías, desde paquetes (IP/MPLS) hasta recursos ópticos, lo que permite ofrecer una gran variedad de servicios que dan soporte a las múltiples características y demandas de las distintas aplicaciones. ABNO opera para dar servicio a estos requisitos de demanda, proporcionando los siguientes tipos de servicio a las aplicaciones:

- Programar y establecer las conexiones necesarias para satisfacer los requisitos de las distintas conexiones entre aplicaciones. Las conexiones pueden incluir conexión de paquetes (en el caso de MPLS-TE) o LSPs ópticos (en el caso de GMPLS).
- Optimizar el retardo de los flujos de tráfico en la red, incluyendo la compartición de archivos, transmisión de video (media streaming) y comunicaciones en tiempo real descritas como Application Layer Traffic Optimization (ALTO) [6].
- Control remoto de los componentes de red permitiendo gestionar de manera coordinada los recursos a través de técnicas como el OpenFlow [7], y la Interface to the Routing System (I2RS)[8].
- Interconexiones de Content Delivery Networks (CDNi) [9] a través del establecimiento y reajuste de las conexiones entre las redes de distribución de contenidos.
- Coordinación de los recursos de la red.
- Redes virtuales privadas (VPN).

2.2. Componentes de la arquitectura ABNO

La arquitectura ABNO combina una serie de componentes y mecanismos que hacen posibles los siguientes procedimientos y funcionalidades:

- Políticas de control de las entidades y aplicaciones para gestionar las peticiones que requieren el uso de recursos de red o de conexiones establecidas sobre ella.
- Reunir información sobre los recursos disponibles en la red.
- Reunir información sobre los recursos de red multicapa y cómo se distribuyen las topologías.
- Manejo de las peticiones y las respuestas de cálculos de caminos.
- Provisión y reserva de recursos de la red.
- Verificación de las conexiones y los recursos de instalación.

Como ya se ha mencionado, la arquitectura ABNO está compuesta por una serie de componentes, protocolos y procedimientos bien definidos. Describe cómo estos elementos funcionales, pertenecientes a

un gran número de tecnologías separadas, y estos protocolos, que han sido desarrollados a través de los años, pueden ser combinados en una única arquitectura para dar servicio a los nuevos requisitos de demanda.

En la Fig. 1 se muestra la arquitectura ABNO.

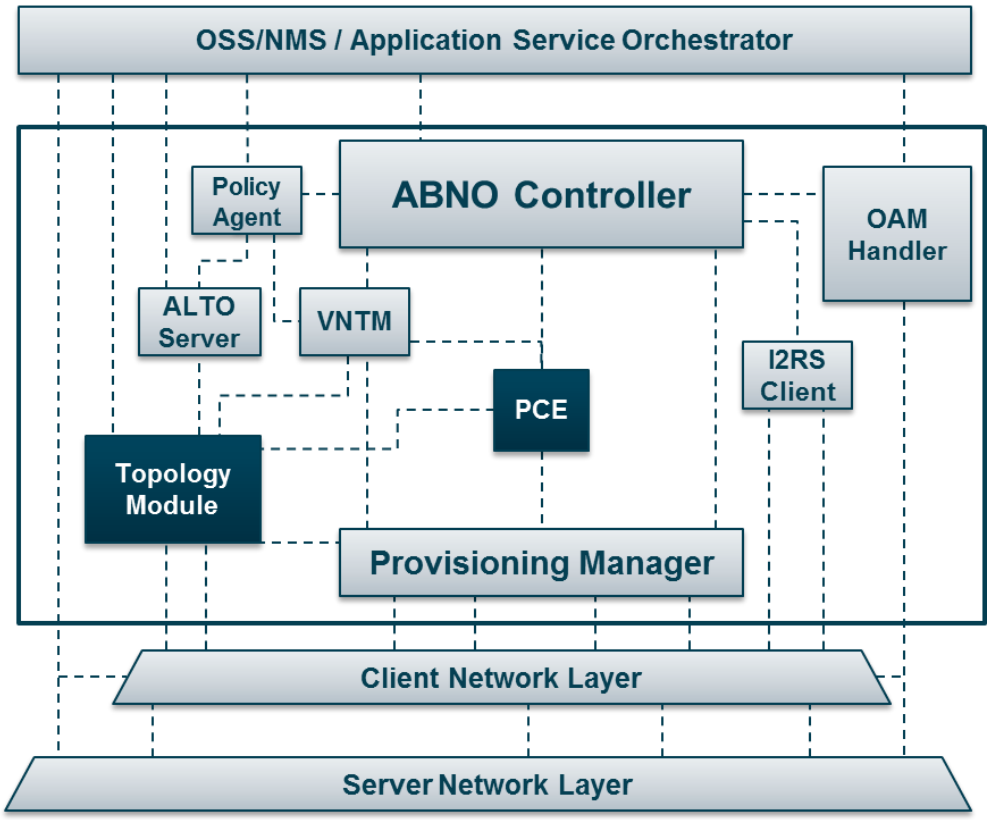

Fig. 1 Adaptive Network Manager Architecture basado en la arquitectura ABNO.

En las siguientes sub-secciones se describen los componentes funcionales que forman la arquitectura ABNO y las interacciones entre ellos.

2.2.1. NMS and OSS

Tanto el Network Management System (NMS) como el Operations Support System (OSS) se usan para controlar, operar y gestionar una red. En la arquitectura ABNO, tanto el NMS como el OSS pueden dirigir peticiones de servicio de alto nivel al controlador ABNO. También pueden establecer políticas que rijan el comportamiento de los componentes en la arquitectura.

El NMS y OSS tienen acceso a los eventos de la red comunicados a través del OAM Handler y por lo tanto, pueden tomar medidas, mostrarlos a los usuarios y dar alarmas en caso de considerarlo necesario. También pueden tener acceso a la base de datos de ingeniería de tráfico (TED) para mostrar a los usuarios el estado de la red.

Por último, el NMS y el OSS utilizan una interfaz para interaccionar con los elementos de red.

2.2.2. Application Service Coordinator

Una aplicación puede ser un programa que corre en un host o servidor y que provee servicios a un usuario, como por ejemplo, una aplicación de videoconferencia. También, una aplicación puede ser una

herramienta de software con la que el usuario hace peticiones de red para establecer servicios específicos, como por ejemplo, conectar dos nodos en la red (conexión punto a punto) o programar reservas de ancho de banda. Por último, una aplicación puede ser un sofisticado sistema de control responsable de organizar la provisión de un servicio de red más complejo, como una red privada virtual (VPN).

En esta arquitectura, todos estos conceptos de aplicación están agrupados en lo que se conoce como Application Service Coordinator, ya que todos ellos, en cierta medida, son responsables de coordinar la actividad de la red para proveer servicios para las aplicaciones. En la práctica, la función del Application Service Coordinator puede estar distribuida en múltiples aplicaciones o servicios.

El Application Service Coordinator se comunica con el controlador ABNO para hacer peticiones de operaciones sobre la red.

2.2.3. ABNO Controller

El controlador ABNO es la principal puerta de acceso a la red para el NMS, el OSS y el Application Service Cordinator. Este controlador regula el comportamiento de la red en respuesta a cambios en las condiciones de ésta y en consecuencia a requerimientos y a políticas de las aplicaciones de red. Es el punto de unión e invoca a los componentes implicados en el orden requerido.

2.2.4. Policy Agent

El Policy Agent juega un papel importante en el control y gestión de la red. Influye significativamente en cómo operan los principales componentes dentro de la arquitectura ABNO.

La Fig. 1 muestra al Policy Agent como un componente que es configurado por el NMS o el OSS con las políticas que aplican. Es el responsable de propagar estas políticas al resto de componentes del sistema.

Aunque se muestre en la Fig. 1 que el Policy Agent solo interactúa con el controlador ABNO, el ALTO Server y el Virtual Network Topology Manager (VNTM), también interactúa con otras componentes y elementos de la red.

2.2.5. OAM Handler

El controlador de Operaciones, Administración y Mantenimiento (OAM) juega un papel importante a la hora de decidir cómo opera una red, detectando errores y tomando las acciones necesarias para reaccionar frente a los problemas que puedan surgir en ella.

En la arquitectura ABNO, el controlador OAM es el responsable de recibir las notificaciones acerca de los problemas potenciales y de mandar a los componentes del sistema implicados que tomen las acciones necesarias para preservar o recuperar los servicios. El controlador OAM también reporta los problemas de la red, y en particular, los problemas de servicio que afectan al NMS, OSS y al Application Service Coordinator.

Además, el controlador OAM interactúa con los dispositivos de red para iniciar acciones de mantenimiento y administración con el plano de datos, como monitorizar o testear.

2.2.6. Path Computation Element (PCE)

El Path Computation Element (PCE) se introduce en la RFC 4655 [2]. Es un componente funcional que computa rutas por la red en respuesta a peticiones. En esta arquitectura el PCE puede recibir las peticiones del controlador ABNO, del VNTM, o simplemente de los elementos de la red.

El PCE calcula sus rutas en base a la topología de red que está almacenada en la Traffic Engineering Database (TED). Existen dos arquitecturas PCE, *stateless* y *stateful*. La arquitectura *stateful* PCE proporciona un cálculo de rutas más sofisticado, ya que guarda información acerca de los LSPs computados previamente que están operativos en la red como se muestra en [2] y [11]. En la arquitectura *stateless* la

TED solo almacena el grafo de la topología de red junto con información de ingeniería de tráfico actualizada por los mecanismos de enrutamiento.

Varios PCEs pueden colaborar para proporcionar caminos en un escenario de multidominio o en redes multicapa. Estos PCEs calcularían rutas basándose en TEDs locales. El PCE es el componente principal de la arquitectura ABNO, por lo que se explicará más en detalle en la sección 2.3.

2.2.7. Topology Module

La arquitectura ABNO incluye bases de datos que contienen información útil para el sistema. Las dos bases de datos principales son la Traffic Engineering Database (TED) y la LSP Database (LSP-DB).

En este trabajo se ha desarrollado este módulo que permite crear y exportar la TED usando distintas tecnologías de red tal y como se explicará en la sección 5.

2.2.8. ALTO Server

El protocolo ALTO está definido como un servicio que ofrece información de red a la capa de aplicación basado en grafos de las capas inferiores. Esta información proporciona una vista simplificada, pero útil para dirigir el tráfico por la capa de aplicación. Los servicios del ALTO permiten compartir información sobre las localizaciones de las redes y los costes entre ellas. El criterio de selección para elegir dos localizaciones depende de información de ingeniería de tráfico como el máximo ancho de banda, mínimo tráfico entre los dominios, menor coste para el usuario, etc.

2.2.9. Virtual Network Topology Manager (VNTM)

Una Topología de Red Virtual (Virtual Network Topology - VNT) [12] es un conjunto de uno o más LSPs que están establecidos en una o más capas de red y que dan lugar a la creación de la Topología de Red Virtual sobre una capa superior. En este entorno el VNTM proporciona información para la gestión eficiente de caminos en una capa de red superior. Por ejemplo, un conjunto de LSPs en una red de multiplexación por división de onda (Wavelength-Division Multiplexing, WDM) puede proporcionar conectividad como enlaces virtuales en una capa superior de conmutación de paquetes. Además, [13] describe cómo el VNTM puede instanciar conexiones entre la capa servidor y la capa cliente. En la arquitectura ABNO, la creación de nuevas conexiones puede ser delegada al Provisioning Manager como será explicado en la sección 2.2.10.

El VNT mejora los enlaces físicos de la capa superior y es configurado estableciendo o eliminando los LSPs en la capa inferior, y anunciando los cambios a la capa superior. El VNT puede adaptarse a las demandas de tráfico, así los recursos de la capa superior pueden crearse o eliminarse según se necesite. Al eliminar recursos virtuales que no se necesitan, se dejan disponibles los recursos en la capa inferior para otros usos.

La creación de una topología virtual no es tarea fácil, se deben tomas decisiones sobre qué nodos de la capa superior son los más adecuados para establecer la conexión y cómo encaminar los LSPs en la capa inferior. Además, son necesarias acciones específicas para poder establecer los LSPs en la capa inferior y que se pueda crear la conexión en la capa superior.

Todas estas acciones y decisiones están influenciadas por la política, por lo que el VNTM que las gestiona, toma parámetros de entrada del Policy Agent. El VNTM también está asociado con el PCE de la capa superior de red y con cada PCE de capas inferiores.

2.2.10. Provisioning Manager

El Provisioning Manager es el responsable de dirigir las peticiones para el establecimiento de los LSPs. Este establecimiento lo puede hacer mandando instrucciones hacia el plano de control o programando

directamente los dispositivos de la red. En este último caso, el Provisioning Manager actúa como un OpenFlow Controller.

Estas operaciones pueden llevase a cabo en dos niveles:

- Peticiones para configurar recursos específicos en los datos o envíos.
- Peticiones que desencadenan una serie de acciones que serán programadas con la asistencia del plano de control.

2.2.11. Client and Server Network Layers

Las capas de cliente y servidor se muestran en la Fig. 1 para notificar que la arquitectura ABNO puede ser usada para coordinar los servicios a través de múltiples redes donde las capas inferiores proporcionarán conectividad a las capas superiores de la red.

Estas capas representan las distintas regiones administrativas y tecnológicas. Es preferible coordinar el control de los recursos de la red y su utilización (es decir, controlar las distintas capas) que controlar y optimizar el uso de recursos para cada capa independientemente. Esto facilita la eficiencia y automatización de la red.

2.3. Arquitectura del Path Computation Element

El Path Computation Element (PCE) es una entidad (componente, aplicación o nodo de red) encargada de calcular caminos o rutas en la red basándose en el grafo de red y aplicando restricciones en la computación en caso de ser requeridas.

El PCE puede formar parte del sistema de gestión de red (Network Management System, NMS) de tal forma que dada una petición de servicio, el NMS se encarga de pedir el cálculo de una ruta al PCE. Éste requiere de la información de estado de la red, almacenada en la base de datos de ingeniería de tráfico (TED). Una vez calculada, el PCE la devuelve al NMS que la envía a los elementos de red para configurar el servicio. En la Fig. 2 se muestran los componentes de esta arquitectura.

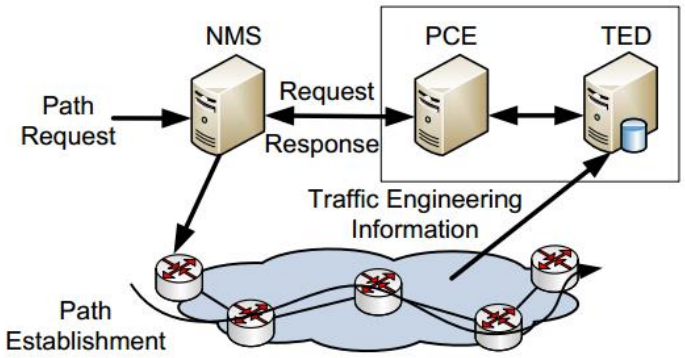

Fig. 2 Path Computacion Element como parte del plano de gestión de red [15].

Por otro lado, el PCE puede también estar situado en el plano de control en lugar de en el plano de gestión. En este caso, existe un Path Computation Client (PCC) que es la entidad encargada de pedir los caminos necesarios al PCE. Normalmente este PCC es un Generalized Multi-Protocol Label Switching (GMPLS) router, capaz de calcular caminos usando el algoritmo de encaminamiento estándar de GMPLS. Pero en caso de que las rutas requieran algo más de complejidad y no sea suficiente con el algoritmo estándar, el PCC pedirá el camino al PCE con las restricciones y condiciones requeridas. El PCE puede estar situado junto al PCC o pueden estar en distintos servidores. Ambas configuraciones se muestran en la Fig. 3.

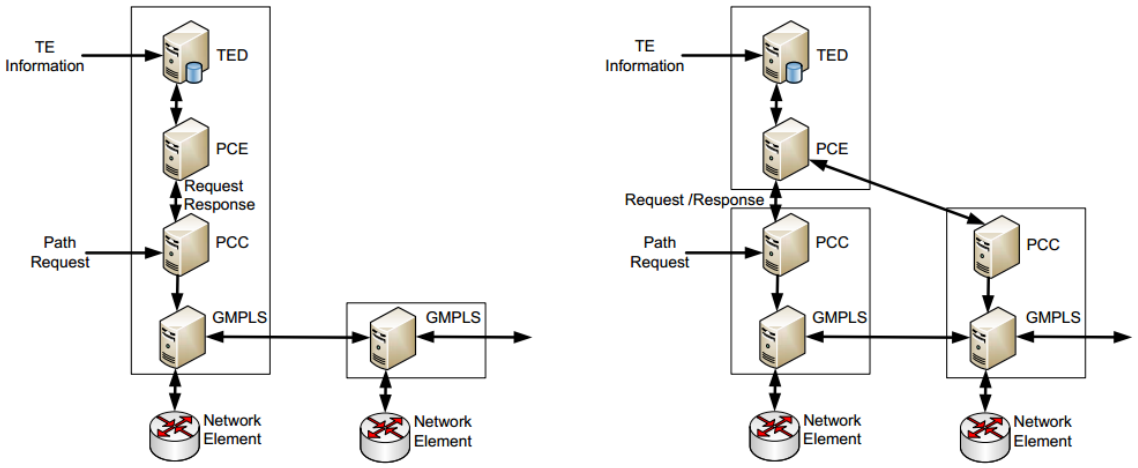

Fig. 3 A la izquierda se muestra el PCE en el mismo servidor que el PCC y a la derecha el PCE y PCC en distintos servidores [15].

La primera configuración es fácil de implementar y no necesita un protocolo de cliente/servidor para la comunicación entre el PCE y PCC. Por otro lado, la configuración en la que el PCC y PCE se encuentran en distintos servidores requiere un protocolo de cliente/servidor estándar que permita la comunicación entre ambos. Este protocolo se denomina Path Computation Element Protocol (PCEP) y opera sobre el Transmission Control Protocol (TCP) [16], [3], el cual satisface los requisitos para una comunicación segura y fiable. Hay siete posibles mensajes en este protocolo: OPEN, KEEPALIVE, REQUEST (PCReq), RESPONSE (PCRep), NOTIFY (PCNtf), ERROR (PCErr) y CLOSE. Estos mensajes se usan en cada una de las distintas situaciones que pueden darse durante la sesión: Session creation, Session keepalive, Path computation, Notification messages, Error messages y Session close [3].

Un PCC puede tener varias sesiones abiertas con distintos PCEs y al revés, un PCE puede tener varias sesiones con varios PCCs.

Para que el PCC y el PCE puedan comunicarse, el PCC debe saber dónde se encuentra el PCE. Esta localización puede ser configurada directamente en el PCC o éste descubrir al PCE. En este trabajo la localización se configura en el PCC. Para comenzar una sesión es necesario que el PCE y el PCC abran una sesión TCP y sobre ella, una sesión PCEP en la que se negociarán los parámetros de la misma. Éstos parámetros se envían en los mensajes de Open e incluyen el Keepalive timer, el Dead timer, etc. Existen dos posibilidades, que el PCC esté permanentemente conectado al PCE o que sólo se conecte si hay una petición. Dependiendo del número de peticiones, una solución será más adecuada que la otra.

Una vez la sesión es establecida, el PCC puede comenzar a enviar los mensajes de petición de camino (PCReq) al PCE que le responderá con un mensaje Response (PCRep) que contendrá la ruta calculada o, si esto no fuese posible, contendrá uno o varios objetos que indiquen los motivos por los que la ruta no pudo ser calculada.

Cuando el PCE o el PCC desea terminar la sesión, envía al otro un mensaje Close y luego cierra la sesión TCP. Si la sesión PCEP la termina el PCE, el PCC tendrá que eliminar todas las peticiones pendientes a enviar al PCE. De igual forma, si el PCC termina la sesión, entonces será el PCE el que elimine todas las peticiones pendientes de calcular que fueron demandadas por dicho PCC. Si la sesión TCP falla, entonces la sesión PCEP se terminará inmediatamente.

Para este trabajo se ha tomado configuración donde el PCE y el o los PCCs se encuentran en distintos servidores y se comunican mediante el protocolo PCEP.

2.4. Conclusiones

La arquitectura ABNO aúna varios componentes ya existentes y otros nuevos componentes que se encargan del cálculo y establecimiento de rutas para el tráfico en la red. Este proceso es automático, y con ello, se consigue disminuir el tiempo de establecimiento de conexiones en la red.

El PCE es el elemento central de la arquitectura, ya que es el encargado del cálculo de caminos pudiendo aplicar, si fuese necesario, ciertas restricciones en base a políticas programadas en la arquitectura. El PCE necesita de un módulo de topología para computar sus rutas que tiene que estar actualizado para que no haya bloqueos a la hora de establecer las conexiones. Es por ello, que en este trabajo se estudia en detalle estos dos módulos de la arquitectura.

En el siguiente capítulo se define un escenario de restauración masiva donde el PCE tiene que calcular las rutas para restablecer el tráfico de un enlace que se ha caído. Se introducirá el mecanismo de pre-reserva que implementa el PCE para conseguir disminuir el número de bloqueos.

3. Mecanismo de Pre-Reserva en entornos con restauraciones masivas

En esta sección se define el mecanismo de Pre-Reserva y se evalúa los beneficios que aporta para el caso de restauración masiva en la red. En primer lugar, se explica el mecanismo de PR en profundidad y las extensiones en el protocolo PCEP necesarias para incluirlo. A continuación, se exponen dos tipos de mecanismos para la recuperación ante fallos de red, mecanismo de protección y de restauración. Finalmente, se define el escenario usado y se muestran los resultados obtenidos tras la emulación.

3.1. Pre-Reserva en el PCE

El PCE calcula sus rutas basándose en información de estado de la red que está almacenada en la base de datos de Ingeniería de Tráfico (TED). Ésta se actualiza mediante la información de los recursos de red que transportan los mensajes de los protocolos de enrutamiento, como por ejemplo, el Open Shortest Path First (OSPF) [17] y sus extensiones de Ingeniería de Tráfico (TE). Como se ha explicado anteriormente, un PCE puede ser *stateful* o *stateless*. En el primer caso, la TED almacenaría, además de los recursos de red, el conjunto de los caminos computados y los LSPs activos en ese momento. Sin embargo, la sincronización y el mantenimiento de un *stateful* PCE puede ser costosa debido a que es posible que existan más elementos que computen caminos a parte del PCE, por lo que necesitarían tener una coordinación entre todos estos elementos para conseguir tener actualizada la TED.

Por otro lado, un *stateless* PCE no contiene la información de los caminos establecidos es su TED y cada petición o conjunto de peticiones es calculada independientemente de las demás. El *stateless* PCE actualiza su TED local con la información de los protocolos de enrutamiento. Esta actualización puede llevar cierto tiempo, lo que hace que la TED esté desactualizada con respecto al estado actual de la red. Este tiempo puede deberse a:

- Las latencias del plano de control, que pueden incrementarse debido a:
    a) El tiempo que tarda el PCC en obtener los caminos computados por el PCE.
    b) El tiempo de establecimiento del camino.
    c) El tiempo que tarda el PCE en actualizar su TED.
- Los protocolos de enrutamiento, que pueden enviar con tiempos fijos los mensajes de actualización, para evitar sobrecargar el plano de control.
- Las peticiones concurrentes que llegan antes de que la TED sea actualizada correctamente. Esto puede darse sobre todo en casos de restauración, cuando un enlace ha fallado y múltiples rutas que estaban establecidas por ese enlace tienen que ser reubicadas en la red en un corto período de tiempo.
- La conexión local del PCE, es decir el PCE tiene que dar servicio tanto a las peticiones de computación de caminos y como a los mensajes de actualización. Por lo tanto, se necesita sincronización al acceder a la TED local del PCE.

Por lo tanto, en el caso de un *stateless* PCE existe cierto riesgo de que las rutas sean calculadas basándose en una información de red desactualizada, por lo que pueden estar formadas por recursos de red ya ocupados previamente por otras y que no llegaron a actualizarse a tiempo en la TED local del PCE. Este error se detecta en el PCC cuando el plano de control intenta establecer el camino y comprueba que los recursos ya están ocupados. Esto provoca que el PCC tenga que pedir de nuevo la petición, es decir, haya reintentos y por lo tanto aumente el tiempo de establecimiento del camino. Los autores de [4] denominan este problema como bloqueo por "robo de lambda". En [5] se propone un mecanismo de Pre-Reserva de recursos como solución a este problema en *stateless* PCEs.

Básicamente, el mecanismo de PR permite al PCE reservar los recursos de una respuesta recién computada por un cierto período de tiempo, denominado Tiempo de Reserva de Recursos (*Tresv*).

Cuando el PCE calcula la ruta, marca los recursos que utiliza dicha ruta como reservados en su TED durante el período de tiempo indicado por *Tresv,* por lo que en los siguientes cómputos de rutas no utilizará esos recursos, evitando posibles bloqueos. Si el establecimiento del camino fue satisfactorio, los mensajes de enrutamiento actualizarán la TED y el PCE eliminará el estado de Pre-Reserva para esos recursos. En caso contrario, una vez expira el *Tresv*, los recursos se vuelven a marcar como libres y podrán volver a ser utilizados para posteriores cómputos.

Para poder llevarse a cabo este mecanismo de PR, tanto el PCC como el PCE deben cumplir una serie de requisitos.

3.1.1. Requisitos del PCC

Cuando el PCC realiza una petición al PCE (PCReq) debe poder indicarle:

- Si desea que el PCE reserve los recursos del camino que calcule.
- La cantidad de tiempo por los que los desea reservar (*Tresv*).
- El tipo y granularidad de los recursos que van a ser reservados. El tipo se refiere a los recursos reservados como ancho de banda, longitud de onda, fibra,… La granularidad es la posibilidad de no solo reservar los recursos computados para el camino sino también los nodos, enlaces o SRLGs asociados.

Además, el PCC debería poder pedir la cancelación de una reserva activa.

3.1.2. Requisitos del PCE

El PCE debe ser capaz de:

- Aplicar políticas tanto si la petición necesita reserva de recursos como si no.
- Computar uno o más caminos de acuerdo con los parámetros indicados en la petición, y basándose en las indicaciones del PCC, evitando que los recursos reservados sean usados en posteriores cómputos durante el *Tresv*.
- El tiempo de reserva debería no ser menor que el tiempo especificado por el PCC. Pero esto no evita que si está configurado por política, el PCE pueda limitar el tiempo de reserva a un tiempo menor al indicado. También podría responder al PCC con un mensaje ERROR, explicando la causa.
- El PCE puede decidir si aplicar una granularidad distinta para la petición que requiera reserva (por ejemplo, reservar una longitud de onda pero no enlaces TE). En estos casos, el PCE debe responder con la reserva que se realizó.

El PCE debería poder responder en el mensaje PCRep lo siguiente:

- Si los recursos han sido efectivamente reservados o bloqueados y por cuánto tiempo se reservarán en aquellos casos en los que el tiempo de reserva especificado por el PCC no coincida con el tiempo reservado.
- La granularidad de la reserva, que podría ser distinta de la demandada por el PCC.
- Proporcionar un medio que permita al PCC cancelar una reserva activa (por ejemplo, un identificador de la reserva).

3.2. Extensiones al protocolo PCEP

En [5] se proponen una serie de extensiones al protocolo PCEP para permitir el mecanismo de PR de recursos. Estas extensiones permiten al PCC indicar en la petición por cuánto tiempo (*Tresv*) desea que el

PCE reserve los recursos del camino computado para que en posteriores peticiones, el PCE no use aquellos recursos que tenga reservados.

Estas extensiones reúnen todos los requisitos previamente mencionados.

3.2.1. Petición de reserva de recursos

El PCC solicita al PCE la reserva de los recursos incluyendo un objeto de reserva (RESERVATION object) en el mensaje PCReq. Éste objeto es opcional, es decir, si no se desea reservar los recursos no es necesario incluirlo.

La estructura del mensaje PCReq sería la siguiente:

```
<PCReq Message> ::=   <Common Header>
                      [svec_list]
                      <request-list>
```

Donde,

```
<svec-list> ::= <SVEC>
                [<OF>]
                [<GC>]
                [<XRO>]
                [<metric-list>]
                [<PCC-REQ-ID> <RESERVATION>]
                [<svec-list>]

<metric-list> ::= <METRIC>
                [<metric-list>]

<request-list>::= <request>
                [<request-list>]

<request>::=    <RP>
                <END-POINTS>
                [<LSPA>]
                [<BANDWIDTH>]
                [<metric-list>]
                [<OF>]
                [<PCC_REQ_ID> <RESERVATION>]
                [<RRO> [<BANDWIDTH>]]
                [<IRO>]
                [<LOAD-BALANCING>]
```

En la Fig. 4 se muestra el formato del objeto de reserva (<RESERVATION>):

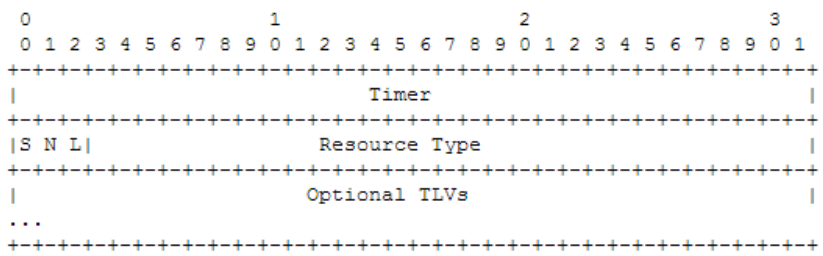

Fig. 4. Formato del objeto de reserva.

- *Timer* indica por cuánto tiempo (en ms) se desea reservar los recursos, codificado como un entero sin signo de 32 bits.
- *Resource Type* indica el tipo de recursos que se van a reservar. Si el valor es 0 significa que se usará los recursos configurados por defecto:
  - Ancho de banda (PSC, L2SC,…)
  - Franja de tiempo (Sonet/SDH TDM)
  - Longitud de onda (G709 OTN OCh or WSON LSC)
  - Tributary slot (G709 OTN ODU-k TDM)
- *Bit L,* si está activo, los TE Links deberían formar parte de la reserva.
- *Bit N,* si está activo, los nodos deberían formar parte de la reserva.
- *Bit S,* si está activo, el conjunto de SRLG (Shared Risk Link Group) debería formar parte de la reserva.

3.2.2. Contestación del estado de la reserva

Se ha extendido el mensaje PCRep para que el PCE pueda indicar que la reserva se ha efectuado. El formato es el siguiente:

```
<PCRep Message> ::= <Common Header>
                    <response-list>
```

Donde,

```
<response-list>::=<response>[<response-list>]

<response>::=<RP>
            [<NO-PATH>]
            [<attribute-list>]
            [<path-list>]

<path-list>::=<path>[<path-list>]

<path>::= <ERO><attribute-list>
```

Donde,

```
<attribute-list>::=[<LSPA>]
                   [<BANDWIDTH>]
                   [<metric-list>]
                   [<IRO>]
                   [<RESERVATION_CONF>]
```

```
<metric-list>::=<METRIC>[<metric-list>]
```

El formato del objeto de confirmación de reserva (<RESERVATION_CONF>) se muestra en la Fig. 5.

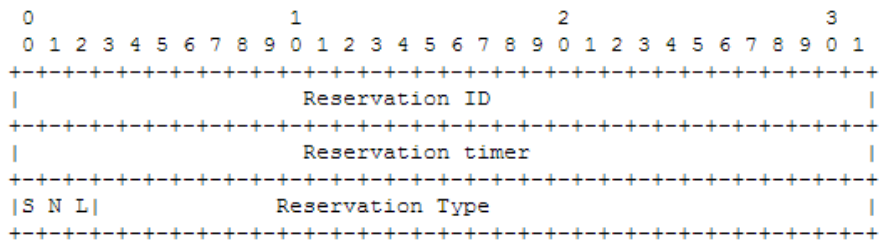

Fig. 5. Formato del objeto de reservation_conf.

- *Reservation ID* es el identificador de la reserva.
- *Reservation Timer* indica el tiempo (en ms) que se reservarán los recursos. El PCE puede decidir si reservar los recursos por un tiempo distinto al indicado por el PCC, por lo que este campo es necesario en esos casos.
- *Reservation Type* es el tipo de reserva.

3.2.3. Cancelación de la reserva

Un PCC puede pedir que se cancele una reserva activa mandando un mensaje de notificación al PCE que incluya el identificador de la reserva.

El mensaje de notificación (PCNtf) debe llevar al menos un objeto de Notificación. Puede llevar varios objetos de Notificación en caso de que el PCC pretenda notificar varios eventos. El formato es el siguiente:

```
<PCNtf Message>::=<Common Header>
                  <notify-list>

<notify-list>::=<notify> [<notify-list>]

<notify>::= <notification-list>

<notification-list>::=<NOTIFICATION>[<notification-list>]
```

El objeto de Notificación usado para cancelar la reserva de caminos debe incluir el identificador de la reserva, que se indica mediante la TLV RESERVATION_ID (Fig. 6).

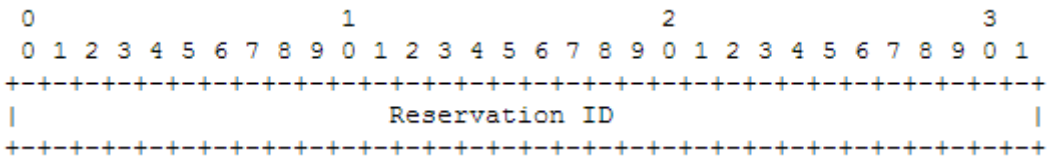

Fig. 6. Formato del objeto de reservation_id.

3.3. Mecanismos de restauración y protección

En las redes puede haber fallos, como la caída de un enlace de la red, el fallo de un nodo de la red, etc. Esto provoca que el tráfico que pasaba por esos elementos de red tenga que ser reubicado. Es muy

importante minimizar la ocurrencia de estos fallos (Mean Time Between Failures*, MTBF*), o al menos, minimizar el tiempo de recuperación ante éstos (Mean Time To Repair*, MTTR*). De hecho, una red 'ideal' con un tiempo entre fallos muy elevado no necesitaría estos mecanismos de restauración o protección ya que prácticamente nunca habría fallos. Pero eso no ocurre en las redes reales, y es por ello que existen distintos mecanismos para solventar estos casos, como son el mecanismo de restauración y mecanismo de protección. Éstos mecanismos no son los únicos que existen, pero sí los más comunes.

El mecanismo de restauración actúa una vez ocurrido el fallo, buscando rutas alternativas por donde desviar el tráfico afectado. Por lo que para este mecanismo es de vital importancia minimizar el tiempo empleado en restaurar todo el tráfico.

El mecanismo de protección actúa antes de que ocurra el fallo, es decir, a la vez que se calcula un camino para transportar el tráfico de datos, se calcula otro camino de respaldo o seguridad por donde se enviará el tráfico en caso de que el primer camino fallase. Este camino de respaldo no comparte ningún recurso con el camino principal, por lo que de haber un fallo en el primer camino, el segundo no tendría en principio porqué fallar. El tiempo empleado para restaurar el tráfico en este caso es prácticamente nulo. La desventaja de este mecanismo reside en que se usan más recursos de los necesarios, ya que se está reservando el doble de recursos para cada conexión (la primaria y la de respaldo).

3.4. Escenario de experimentación

Cuando ocurre un fallo en la red que afecta a varias conexiones (LSPs) es necesario recalcular nuevas rutas para el tráfico afectado. En caso de usar un mecanismo de protección, este paso no sería complicado ya que se desviaría cada LSP por su camino de respaldo que estaría calculado previamente. Sin embargo, en caso de usar el mecanismo de restauración, se tendría que calcular nuevas rutas para todo el tráfico en el menor tiempo posible. Este último mecanismo será nuestro objeto de estudio, es decir, se comprobarán los resultados que ofrece el mecanismo de PR para la restauración masiva de tráfico en la red.

En la restauración masiva el PCE recibe una gran cantidad de peticiones en un corto período de tiempo. Como se ha explicado anteriormente, el PCE basa sus cómputos en la información de estado de la red almacenada en su TED local, que puede estar desactualizada con respecto al estado real de la red.

El escenario utilizado para la simulación es la red 14-node NSFNet mostrada en la Fig. 7. El Round-Trip delay time (RTT) entre el PCC y el PCE es de 10 ms, el tiempo medio que tarda el PCE en computar los caminos es alrededor de 50 ms y el retado para la configuración de ROADM varía entre 10 ms y 400 ms Estos valores son similares a los usados en [18].

El algoritmo que usa el PCE para computar los caminos es el AURE [19]. Este algoritmo crea un grafo de toda la red para cada lambda. Si una lambda está ocupada, entonces se elimina el enlace en el grafo correspondiente a dicha lambda, por lo que en ese grafo será como que no existe conexión entre los nodos conectados por ese enlace. Este algoritmo calcula caminos para todas las lambdas disponibles, es decir aplica Shortest-Path (SP) para tantos grafos como lambdas tenga, y devuelve el camino más corto de todos los calculados y en el caso de que haya varios, devolverá el de la primera lambda disponible (First Fit).

En esta implementación se emula el corte de una fibra, afectando a varios LSPs, concretamente a 28. Se ha utilizado el enlace más cargado. Cuando el nodo origen de cada camino afectado se da cuenta del fallo, envía una nueva petición de camino al PCE. El PCE tiene que calcular caminos para estas nuevas peticiones en un corto período de tiempo, sin poder usar el enlace que dio el error. Esto sólo será posible si existen recursos libres en la red, es decir, si existen enlaces con lambdas vacías para acomodar todas las peticiones.

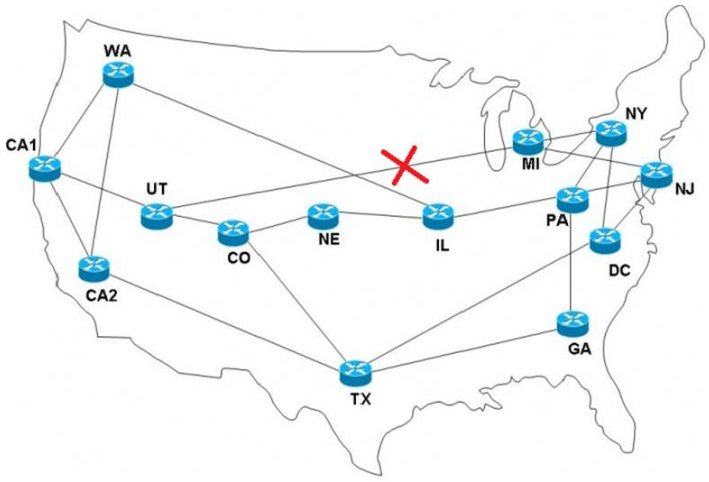

Fig. 7 Red 14-node NSFNet.

3.5. Evaluación de resultados

3.5.1. Impacto del Tiempo de Reserva

Se han realizado varias emulaciones variando el *Tresv* con el objetivo de determinar cómo afecta este parámetro a la hora de calcular los nuevos caminos para aquellos LSPs afectados. En la Fig. 8 podemos ver la probabilidad de que haya al menos un reintento en la petición del camino, es decir, que el primer camino devuelto por el PCE no sea correcto, contenga recursos ya ocupados, y el PCC tenga que volver a pedir un camino, variando el *Tresv* de los recursos. En la misma gráfica se compara también cómo afecta para este bloqueo el tiempo que el ROADM tarda en configurar el enlace.

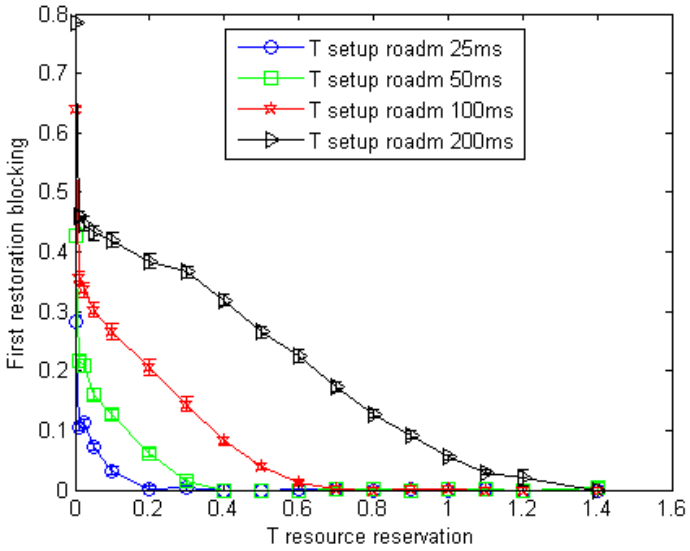

Fig. 8 Probabilidad de bloqueo vs. *Tresv* de recursos (ms).

Como podemos apreciar, a medida que el *Tresv* aumenta, el número de reintentos necesarios para restablecer los LSPs decrece. Si el *Tresv* es suficientemente grande, el bloqueo desaparece, es decir, no habrá bloqueo por robo de lambda y todas las peticiones serán calculadas correctamente en el primer intento.

Si nos fijamos ahora en cómo afecta al bloqueo el tiempo de configuración del ROADM, podemos apreciar que a medida que el tiempo de configuración es mayor, la probabilidad de bloqueo por robo de

lambda aumenta. Esto es razonable ya que cada ROADM tarda más tiempo en configurar y establecer el enlace, por lo que el camino tarda más tiempo en establecerse, y como consecuencia el mensaje de confirmación (OSPF LS Update) llega más tarde al PCE y su TED está durante más tiempo desactualizada. Es por este motivo por lo que se necesita mayor tiempo de reserva de recursos para conseguir menor bloqueo. En el caso de estudio, 200 ms es el mayor tiempo de configuración del ROADM. Para este ROADM se necesita una media de 1,4 segundos de *Tresv* para que los 28 caminos afectados puedan ser acomodados en el primer intento sin que exista bloqueo por robo de lambda.

La Fig. 9 muestra el número de intentos necesarios para conseguir establecer todos los caminos en un escenario donde el PCE no implementa el mecanismo de PR. En este escenario se ha fijado el tiempo de configuración del ROADM a 200 ms. Se puede ver en el histograma que hay una cantidad significativa de reintentos para establecer las peticiones a la primera (alrededor del 80%), y que hay bastantes peticiones con un número elevado de éstos (más de 10 reintentos).

Es otras palabras, es necesario un mecanismo de PR de recursos cuando el problema a tratar es la restauración masiva de rutas, ya que según se ha visto, si el PCE no lo implementa el número de colisiones al establecer los caminos crece y por lo tanto, el número de reintentos para establecer las peticiones se eleva.

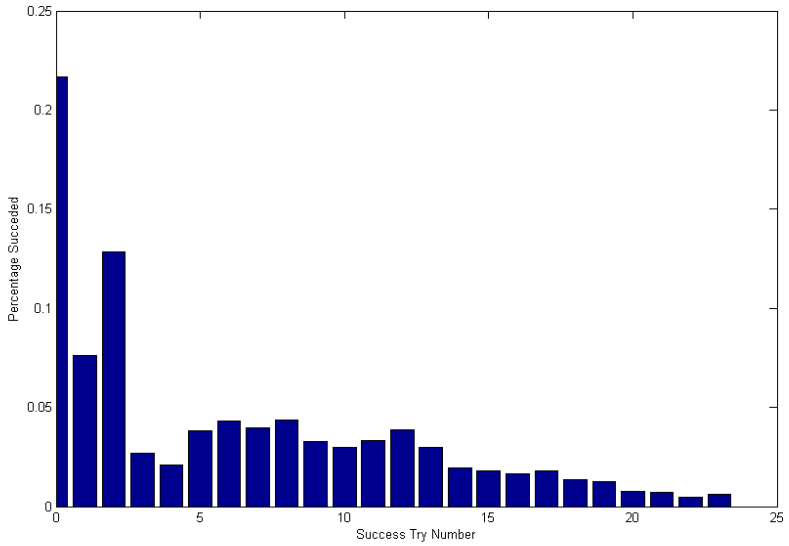

Fig. 9 Número de intentos necesarios para completar la restauración de los distintos caminos afectados usando el algoritmo First Fit.

3.5.2. Rendimiento de la restauración

La Fig. 10 muestra el tiempo medio necesario para restaurar cada LSP afectado por el fallo del enlace cuando el PCE usa el mecanismo de PR con *Tresv* de 1 s y cuando no lo usa. Además, se ha hecho el estudio para distintos tiempos de configuración de ROADM. En vista de los resultados, la PR permite que la restauración sea más rápida y además, a medida que el tiempo del ROADM es mayor, el tiempo necesario para la restauración crece considerablemente.

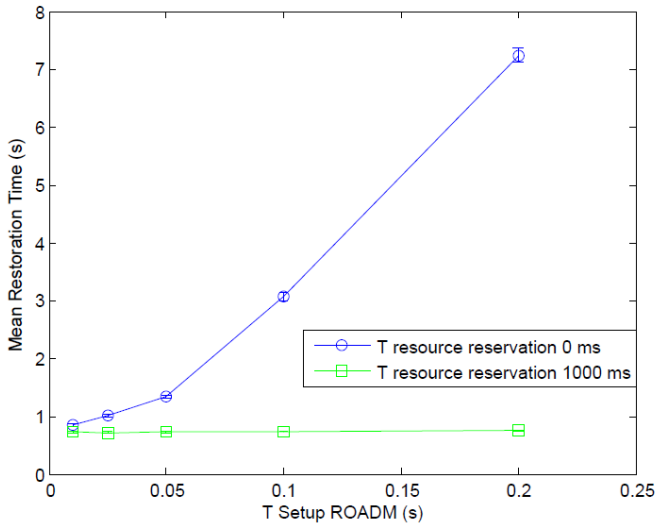

Fig. 10 Tiempo medio requerido restaurar todos los LPSs afectados por el fallo, con y sin mecanismo de PR de recursos para distintos tiempos de configuración de ROADM.

Lo mismo puede verse en la Fig. 11 que muestra el tiempo total necesario para restaurar completamente todas las rutas afectadas por el fallo del enlace. En ella vemos que el uso del mecanismo de PR puede reducir de forma significativa el tiempo de restauración, llegando hasta 18 segundos (con tiempo de configuración ROADM igual a 200 ms) si no se usa PR frente a los 2 segundos cuando se usa la PR con *Tresv* igual a 1 s.

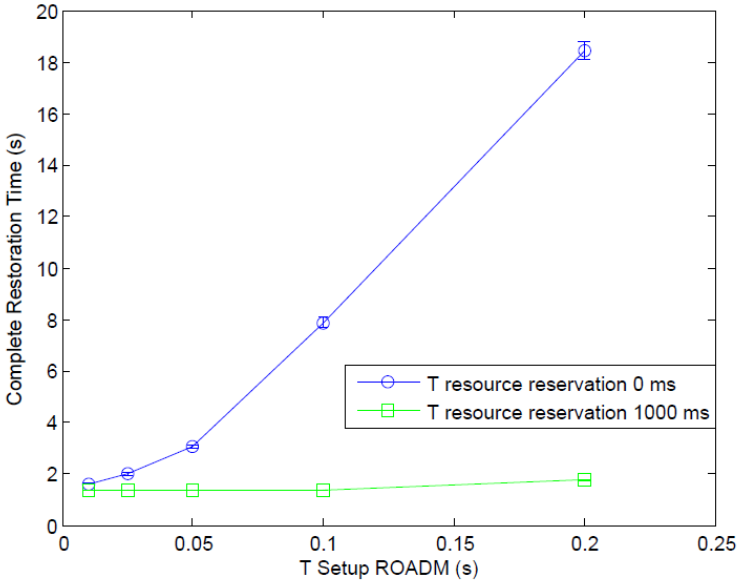

Fig. 11 Tiempo total necesario para restaurar cada conexión afectada por el fallo, con y sin mecanismo de PR para distintos tiempos de configuración de ROADM.

Estos resultados siguen la misma línea que los obtenidos en [18]. En este trabajo se han realizado los experimentos en un entorno real mientras que en [18] era un entorno simulado.

3.6. Conclusiones

El PCE calcula los caminos basándose en la TED local, que puede estar desactualizada con respecto al estado de la red. Esto provoca que el PCE pueda devolver caminos con recursos ya ocupados, dando lugar al bloqueo por robo de lambda. Este bloqueo puede evitarse con el mecanismo de PR de recursos.

El caso de estudio en esta sección es un escenario de restauración masiva de red. Se observa cómo el mecanismo de PR permite eliminar este bloqueo, minimizar el número de reintentos de peticiones y como consecuencia, minimizar el tiempo necesario para restablecer todo el tráfico afectado por el fallo.

En el siguiente capítulo se estudiará este mecanismo de PR en un escenario con múltiples PCEs computando las rutas en un mismo dominio.

4. Mecanismos de Pre-Reserva en entornos de múltiples PCE

En este apartado se define y evalúa el mecanismo de PR en escenarios de múltiples PCEs en un mismo dominio. Se evaluará cómo varios PCEs, usando el mecanismo de PR, se encargan de la gestión de los recursos de red y de la computación de las rutas demandadas en las peticiones.

4.1. Esquemas de Pre-Reserva para múltiples PCEs

En esta sección se explicarán tres escenarios distintos con múltiples PCEs en un mismo dominio de red. Esta clasificación atiende al grado en el que los PCEs tienen conocimiento de su entorno, es decir, del conocimiento que tiene cada PCE de los demás PCEs de su dominio. Se han estudiado tres posibles clasificaciones: PCEs independientes, PCEs encargados de recursos independientes de la red, PCEs colaborativos.

4.1.1. PCEs independientes

En este escenario los PCEs actúan de manera independiente. Ninguno tiene conocimiento de los demás PCEs del dominio. El PCC manda la petición con el mecanismo de PR activo, por lo que, cada PCE computará los caminos y reservará los recursos por el tiempo indicado (*Tresv*). Esta reserva se efectúa en la TED local de cada PCE, por lo que no será compartida ni podrá ser accedida por los otros PCEs del mismo dominio. Este escenario se muestra en la Fig. 12.

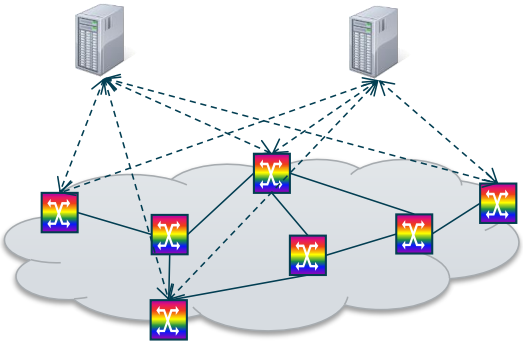

Fig. 12 Dos PCEs independientes en un mismo dominio.

4.1.2. División de los recursos de la red (Lambda sub-set assignment)

Se ha definido otro escenario en el que cada PCE se encarga de un subconjunto de lambdas de la red a la hora de computar los caminos. Este subconjunto es disjunto, es decir, los PCEs no comparten recursos. Al no compartir recursos y usar el mecanismo de PR, se espera que el único bloqueo de lambda posible solo dependa de cada PCE en particular. Esta estrategia puede ser considerada como una división de dominios dentro del mismo dominio, cada sub-dominio con un rango de lambdas, y cada PCE encargado de un único sub-dominio. Esto hace que cada PCE tenga una red más pequeña en cuanto a recursos de red. Este escenario se muestra en la Fig. 13.

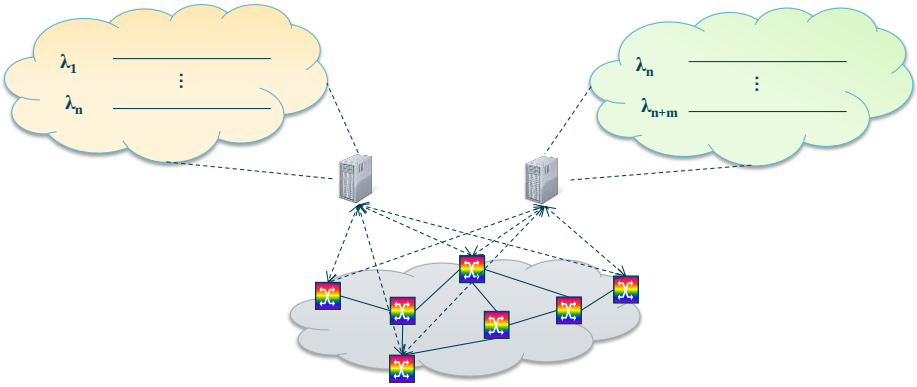

Fig. 13 División de los recursos de la red.

4.1.3. PCEs colaborativos

En este escenario los PCEs comparten los recursos de la red a la hora de computar los caminos, igual que en el caso de PCEs independientes, pero se diferencian en que no comparten la información de los recursos de red reservados. Esto hace que los PCEs del mismo dominio tengan actualizada su TED con sus pre-reservas y las pre-reservas de los demás PCEs. Esta coordinación puede reducir bastante la probabilidad de bloqueo.

La Fig. 14 muestra este escenario con dos PCEs colaborativos, donde se puede ver que entre los dos PCEs hay intercambio de mensajes de notificación.

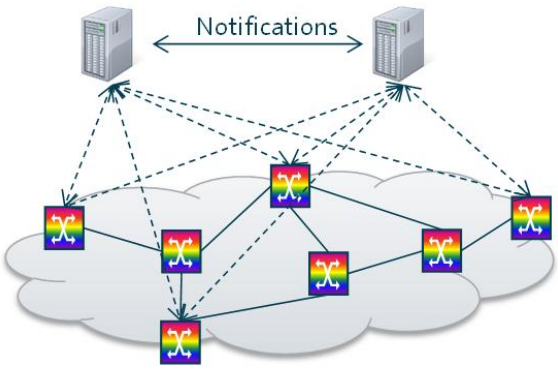

Fig. 14 Escenacio con PCEs colaborativos.

Este intercambio de mensajes podemos verlo más detalladamente en la Fig. 15. En ella el PCC manda un mensaje de petición de camino al PCE1. Éste computa la ruta, manda la respuesta al PCC y un mensaje de notificación al PCE2 que contiene la información de pre-reserva de recursos del camino que fue computado. El PCE2 evalúa este mensaje y reserva los recursos en su TED local por el tiempo indicado en el mensaje de notificación (*Tresv*). El resto del proceso mostrado en la Fig. 15 muestra el establecimiento del plano de control y el proceso de actualización de la ruta.

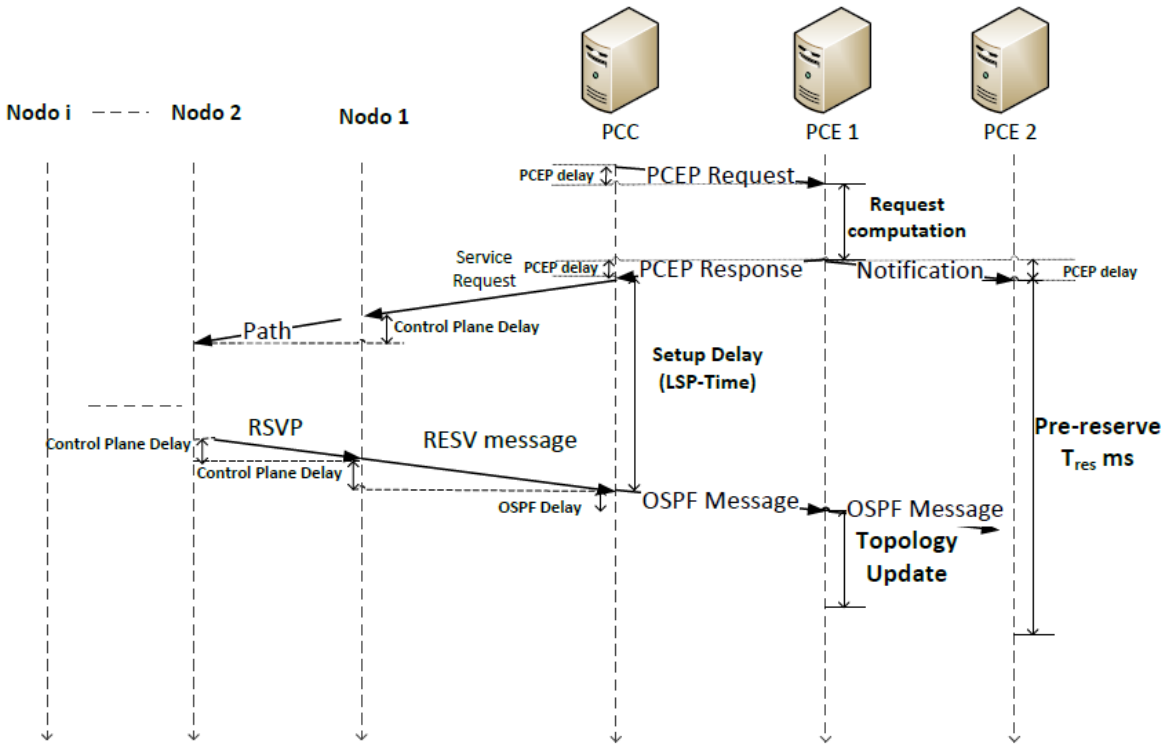

Fig. 15 Flujo de mensajes de pre-reserva en un escenario con dos PCEs colaborativos.

4.2. Comportamiento de los PCCs

No solo los PCEs pueden variar su comportamiento en un escenario de múltiples PCEs, sino que también los PCCs pueden comportarse de distinta forma. Se han definido dos estrategias distintas con respecto al método de elegir el PCE para hacer la petición.

4.2.1. Asignación estática de PCE

En esta estrategia los PCCs están configurados para pedir peticiones a un único PCE. Al principio de la emulación les es asignado un PCE que será al que hagan las peticiones. Teniendo en cuenta que hay varios PCEs, la asignación de PCEs se debe hacer con cierto criterio para evitar que algún PCE se sobrecargue.

4.2.2. Balanceo de carga

También interesante es la posibilidad de implementar un mecanismo de balanceo de carga donde los PCCs hacen peticiones a los PCEs disponibles, sin que les sea asignado un único PCE. Por un lado, esto conlleva cierto grado de flexibilidad ya que los PCCs pueden hacer peticiones a distintos PCEs pudiendo distribuir la pre-reserva de lambda. Por otro lado, el proceso de balanceo de carga es más complicado, ya que el PCC es el encargado de elegir el PCE al que hacer la petición.

4.3. Extensiones al protocolo PCEP

Se han propuesto ciertas extensiones al protocolo PCEP para soportar el mecanismo de Pre-Reserva con PCEs colaborativos. Estas extensiones se han basado en la realizadas en [5] que fueron explicadas en el apartado 3.2.

La primera consiste en cómo indicar que la sesión es de un PCE colaborativo. Entre ambos PCEs se inicia una sesión PCEP para que sea posible el envío y recepción de mensajes de notificación entre los PCEs.

Cuando un PCE está funcionando en el modo colaborativo, tiene que indicar al otro PCE que quiere recibir su información de pre-reserva. Para ello se ha definido un flag en el mensaje OPEN que indicará que los PCEs están actuando en el modo colaborativo.

Para enviar la información de pre-reserva, se usará un mensaje de notificación. Dentro de ese mensaje de notificación, se propone la utilización de un objeto Path Reservation TLV, que incluye un campo con el tiempo de reserva de recursos (parámetro *Tresv*) y el Explicit Route Object (ERO) computado por el PCE. La estructura del objeto Path Reservation TLV se muestra en la Fig. 16. Los primeros 8 bytes de la TLV se han tomado del objeto RESERVATION [5]. Se ha definido un nuevo flag (b) para indicar si el camino es bidireccional, permitiendo con un único mensaje de notificación pre-reservar el camino en ambas direcciones, en vez de tener que mandar dos mensajes de notificación.

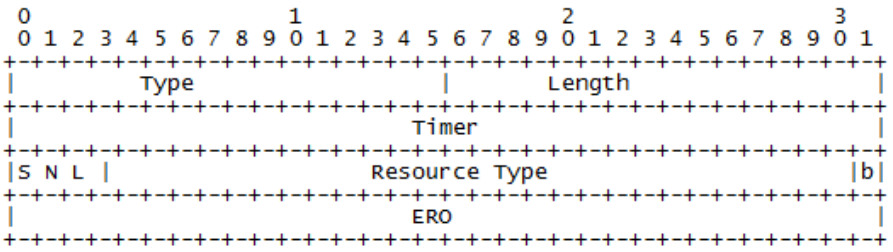

Fig. 16 Path Reservation TLV.

El objeto ERO contiene la lista de direcciones y etiquetas que componen el camino y como se explica en [20], está compuesto por una lista de sub-objetos. El objeto ERO del mensaje de notificación que se propone está compuesto por un conjunto de interfaces no numeradas [21] o prefijos IPv4 para indicar las direcciones del camino y de un objeto DWDM Wavelength Label [22] para indicar la longitud de onda elegida para el camino.

En la Fig. 17 se muestra una captura de Wireshark donde aparece el mensaje de notificación que se propone. El camino puede tener distintas lambdas por enlace, pero en los experimentos se ha asumido que la lambda sea única para todo el camino.

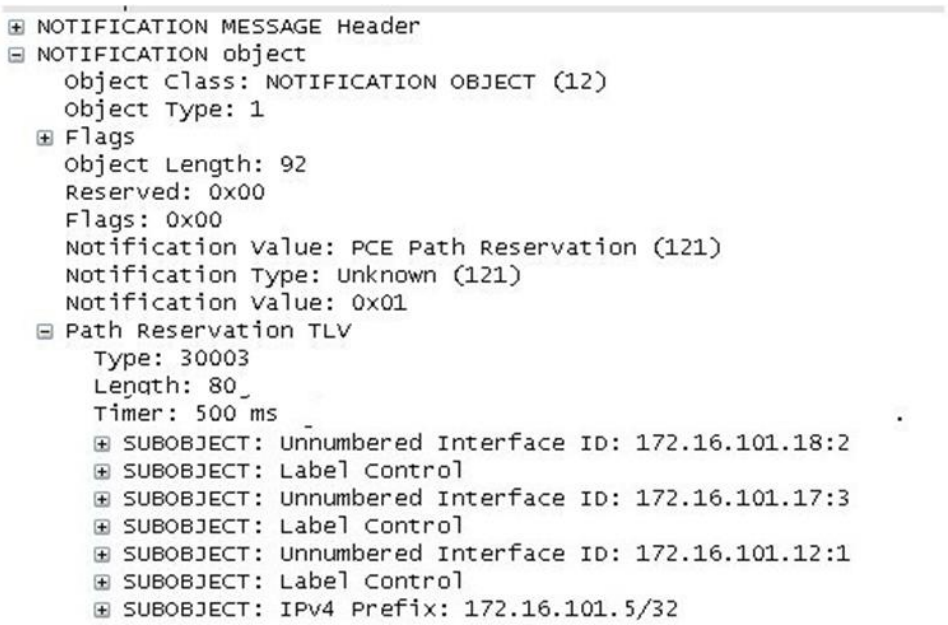

Fig. 17 Captura de Wireshark del mensaje de notificación propuesto.

4.4. Escenario de experimentación

El escenario de experimentación usado en los experimentos es un desarrollo del PCE por TID que reúne todos los estándares definidos por el IETF. La Fig. 18 muestra el escenario usado, donde hay dos PCEs y dos PCCs. Los PCCs comparten el Emulador de Red (Network Emulator - NE) y mantienen una sesión PCEP con los PCEs. Se ha simplificado en el dibujo el número de sesiones PCEP, pero en realidad si los PCCs están funcionando en modo de balanceo de carga, mantendrían una sesión PCEP con cada uno de los PCEs del dominio. Cada PCE tiene su propia TED local con la información de la red. Se ha desarrollado un NE que, desde el punto de vista del PCC, emula el comportamiento de los elementos de la red. Este NE incluye una TED que refleja el tráfico real en tiempo de ejecución que hay en la red. Esta separación entre las TEDs de los PCEs y la del NE permite tener el mismo comportamiento en este caso experimental que el que habría en una red real.

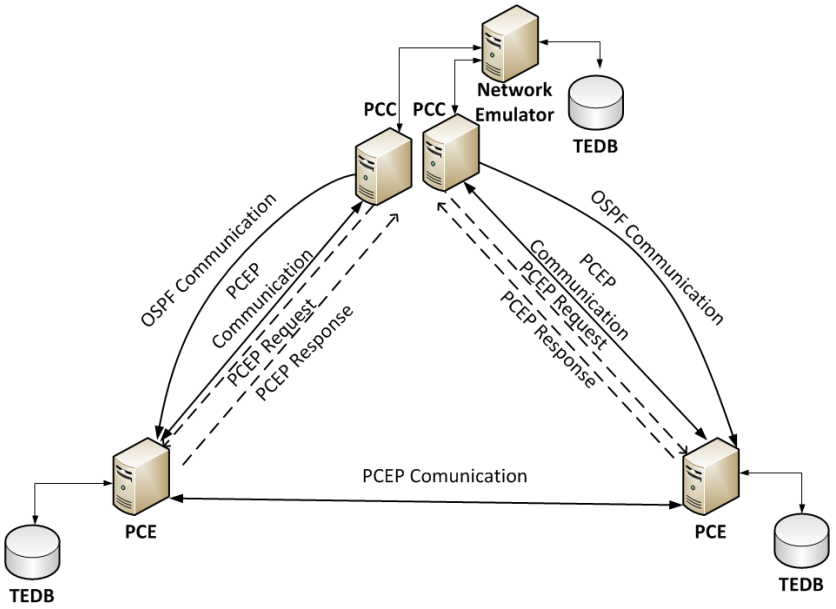

Fig. 18 Escenario experimental.

La TED del PCE se actualiza mediante el NE usando la información de los protocolos Interior Gateway Protocol (IGP); en particular, los mensajes de actualización serán los Link State (LS) Update del protocolo OSPF. El proceso de actualización se hace dinámicamente cuando los recursos de red son reservados en el NE. Éste emula el retardo existente en el plano de control de los enlaces entre los nodos. La Fig. 15 muestra el establecimiento de un LSP en el NE (Nodes 1, 2,…i) y este retardo del plano de control causado por la transmisión del camino entre los nodos.

Un ejemplo del escenario de experimentación comienza por crear una TED para el NE con las conexiones entre los nodos como enlaces de fibra óptica compuestos por longitudes de onda disponibles. Un PCC pide un camino a un PCE y la respuesta recibida es procesada en el NE. Como ya se explicó anteriormente, cada respuesta se compone de un objeto ERO que lleva la información de los enlaces y longitudes de onda de la ruta. Para cada enlace, el NE comprueba en su TED si está disponible esa longitud de onda. En caso de que esté disponible, la TED se actualiza poniéndola como ocupada en ese enlace, y espera el tiempo decidido como retardo del plano de control. Cuando la respuesta está correctamente establecida, el NE envía al PCE un mensaje OSPF LS Update por cada enlace establecido. En caso de que la longitud de onda estuviera ocupada para algún enlace del camino, el NE para el proceso establecimiento del camino, des-reserva los enlaces que estuvieran establecidos para ese camino y advierte al PCC que el camino no pudo ser establecido. Eso significa que hubo un bloqueo por robo de lambda.

En el escenario de experimentación, los PCCs generan tráfico y dependiendo del esquema PCC usado, envían las peticiones a un único PCE o a los dos balanceando las peticiones entre ambos. La carga que se mete en la red es una relación entre el tiempo medio de conexión y el tiempo entre llegadas, por lo tanto es adimensional. El tiempo de conexión es el tiempo que estará ocupado cada enlace. En el caso de estudio el tiempo de conexión se ha fijado en 6 segundos y el tiempo entre llegadas varía para modificar la carga del sistema. PCCs y PCEs están en la misma LAN, por lo que el RTT es menor que 300 ms.

La Fig. 19 muestra el escenario usado para este experimento. Cada enlace está compuesto por 32 canales DWDM wavelength. El algoritmo usado por el PCE para computar los caminos WSON es el K-Shortest Path (KSP) (con K=4) y First-Fit para elegir la lambda disponible.

Fig. 19 Red usada para el caso de múltiples PCEs.

4.5. Evaluación de resultados

4.5.1. Impacto del parámetro Tiempo de Reserva en un escenario con un único PCE

El parámetro *Tresv* debe ser definido basándose en las condiciones de red. Para eliminar el bloqueo por robo de lambda, *Tresv* debe ser lo suficientemente grande para que el establecimiento del mensaje y actualización de la TED del PCE se haga correctamente. Este tiempo engloba tanto el retardo del plano de control en el establecimiento del camino como la diseminación posterior de los mensajes de OSPF que actualizarán la TED del PCE. Sin embargo, el *Tresv* no debe ser mayor que el tiempo de conexión ya que si es más grande entonces se están reservando los recursos incluso cuando la conexión ha acabado, lo que lleva a una mala gestión de los recursos de la red.

Para calcular el *Tresv* adecuado para los experimentos, se ha elaborado un estudio previo en un escenario con un único PCE en el dominio y con una carga fija de 140 Erlangs. En este estudio se han hecho varias emulaciones variando el parámetro *Tresv* hasta conseguir que el bloqueo por robo de lambda desapareciera. En vista de los resultados mostrados en la Fig. 20, no hay bloqueo por robo de lambda cuando *Tresv* es 500 ms. Por lo tanto, se ha elegido *Tresv*=500 ms para los experimentos.

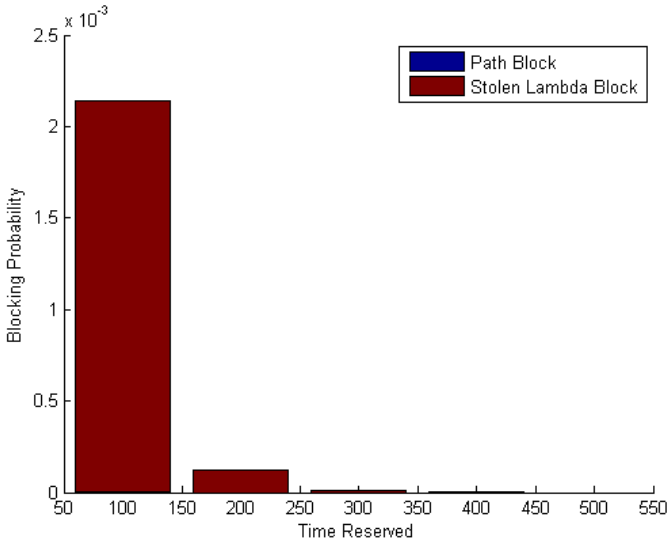

Fig. 20 Probabilidad de bloqueo frente a tiempo de reserva en un escenario de un único PCE y carga de 140 Erlangs.

4.5.2. Comparación de los mecanismos de múltiples PCEs

La Fig. 21 muestra la comparación en términos de probabilidad de bloqueo por robo de lambda obtenidos con las tres arquitecturas de múltiples PCEs por dominio explicadas en la sección 4.1 y con un escenario de un único PCE. El tiempo de reserva está fijo a 500 ms, por lo que no hay bloqueo en el escenario de un único PCE.

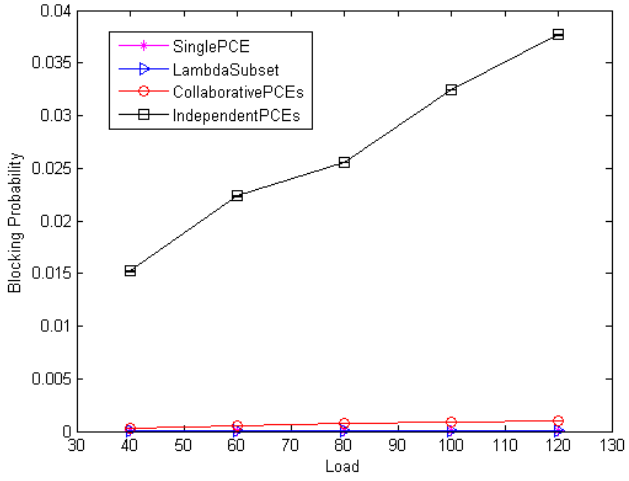

Fig. 21 Comparativa de la probabilidad de bloqueo por Robo de Lambda en un escenario de múltiples PCEs (con balanceo de carga).

Observando la Fig. 21 vemos que el esquema de PCEs independientes obtiene los peores resultados en cuanto al bloqueo por robo de lambda. Esto es lógico ya que cada PCE no comparte con los demás PCEs la información de los recursos reservados, y cada uno computa los caminos en base a su TED local que se actualizará con los mensajes de OSPF una vez se hayan establecido.

Se muestra en la Fig. 22, más en detalle, los tres esquema que ofrecen mejores resultados. En el esquema de división de recursos de la red (*Lambda sub-set*), los PCEs no comparten recursos a la hora de computar los caminos, por lo tanto, el bloqueo es bajo y similar al que se obtiene con un único PCE en el dominio.

Sin embargo, como cada PCE tiene asignados menos recursos de la red que con un único PCE, la probabilidad de bloqueo aumenta para cargas altas, como podemos apreciar en la gráfica.

Finalmente, los PCEs colaborativos reducen el bloqueo por robo de lambda en comparación con los PCEs independientes, ya que la información de reserva de recursos es compartida entre los PCEs. Aun así, el resultado obtenido es peor que con *Lambda sub-set* pero tiene la ventaja de que los PCEs computan sobre todos los recursos de la red en vez de limitarse a solo unos recursos.

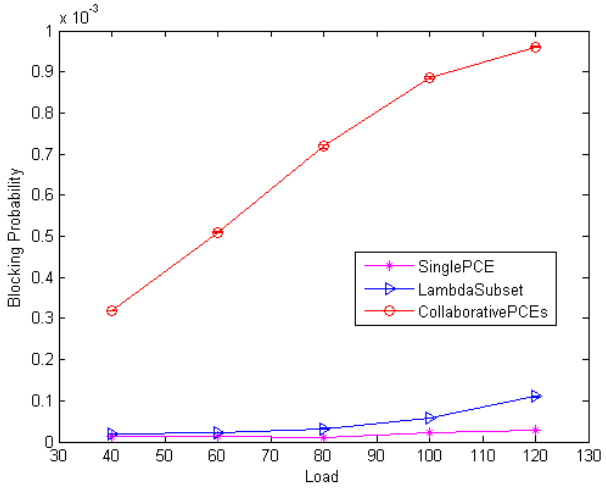

Fig. 22 Comparativa de la probabilidad de bloqueo por Robo de Lambda en un escenario de múltiples PCEs (con balanceo de carga).

En cuanto al tiempo de computación de las rutas en el PCE, se ha visto que para todos los esquemas es prácticamente el mismo, es decir, el tiempo no cambia independientemente del esquema usado. En la Fig. 23 se muestra un histograma con el restado al computar una petición obtenido en uno de los experimentos.

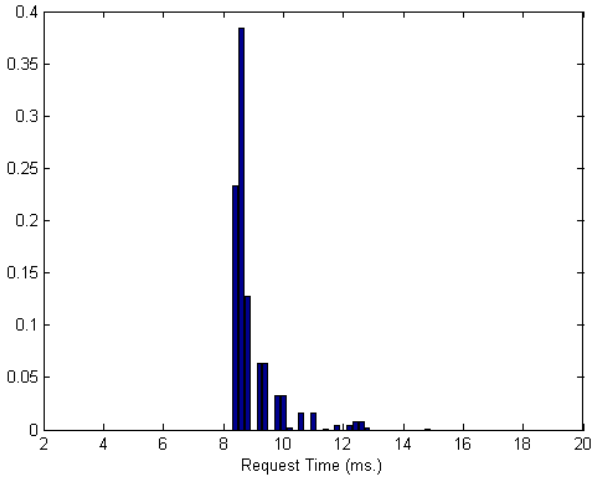

Fig. 23 Histograma del tiempo de cómputo de la petición (ms) en un escenario de PCEs colaborativos para una carga de 100 Erlangs (con balanceo de carga).

4.5.3. Evaluación de las estrategias PCC

El último estudio consiste en medir el rendimiento que ofrecen los distintos esquemas de múltiples PCEs cuando el comportamiento de los PCCs cambia. Como se explicó en la sección 4.2, los PCCs pueden elegir dos estrategias a la hora de hacer las peticiones, elegir siempre al mismo PCE (asignación estática) o elegir al PCE dependiendo de la carga que tenga éste (balanceo de carga). La Tabla 1 muestra la probabilidad de bloqueo por robo de lambda para los escenarios de PCEs independientes y colaborativos con los esquemas de asignación estática de los PCEs y balanceo de carga. La ganancia relativa entre ambos esquemas en el caso del escenario de PCEs independientes es del 12%, pero es menor del 1% cuando la carga aumenta. Por otro lado, el uso de balanceo de carga a la hora de realizar las peticiones a los PCEs ofrece mejores resultados, hasta un 86,1% menos de probabilidad de bloqueo por robo de lambda, con un escenario de PCEs colaborativos. Como las respuestas son compartidas entre ambos PCEs, la información de pre-reserva está repartida más uniformemente en las TED cuando se usa un esquema de balanceo de carga.

PCEs Independientes					
Carga del sistema (Erlangs)	**40**	**60**	**80**	**100**	**120**
Asignación estática PCE	1,39E-02	1,97E-02	2,58E-02	3,22E-02	3,70E-02
Balanceo de carga	1,52E-02	2,24E-02	2,55E-02	3,25E-02	3,77E-02
Ganancia Relativa	-8,9%	-11,7%	0,9%	-0,8%	-1,8%
PCEs colaborativos					
Carga del sistema (Erlangs)	**40**	**60**	**80**	**100**	**120**
Asignación estática PCE	1,60 E-04	4,24E-04	7,04E-04	7,89E-04	1,01E-03
Balanceo de carga	1,84E-04	2,74E-04	3,82E-04	4,24E-04	5,92E-04
Ganancia Relativa	-13%	54,7%	84,3%	86,1%	70%

Tabla 1 Comparación del bloqueo por robo de lambda dependiendo de la estrategia del PCC usada.

4.6. Conclusiones

En esta sección se han estudiado los resultados que ofrece el mecanismo de PR en entornos de múltiples PCEs. Se han estudiado tres escenarios distintos: PCEs independientes, PCEs con división de los recursos de la red y PCEs colaborativos. Para este último escenario se ha extendido el protocolo PCEP para permitir que ambos PCEs se envíen la información de los recursos reservados.

Se ha probado que tanto los PCEs colaborativos como los PCEs con división de recursos de red obtienen mejores resultados en términos de bloqueo por robo de lambda cuando se usa la PR. El primero minimiza este bloqueo gracias al intercambio de mensajes de notificación que contienen la información de los recursos reservados. El segundo tiene un comportamiento similar al que ofrece un único PCE ya que los recursos no son compartidos entre los PCEs. El escenario de PCEs independientes ofrece los peores resultados a pesar de que los PCEs implementen la PR.

En el siguiente capítulo se centrará en el módulo de topología de la arquitectura ABNO.

5. Diseño e implementación de un módulo de topología para la arquitectura ABNO

Tal y como se vio en las secciones anteriores, la arquitectura ABNO permite realizar operaciones de red de una forma automática que van desde el cálculo de rutas para el tráfico en la red y reserva de los recursos necesarios para ellas, hasta la interconexión dinámica de CDNs. Uno de los módulos que tiene la arquitectura ABNO es un módulo de topología, tal y como se puede ver en la Fig. 1. En este trabajo final de master se ha diseñado, implementado y validado el módulo de topología de esta arquitectura que es el responsable de almacenar y proveer información de la topología de red a los módulos internos. En esta sección veremos la definición del módulo y los bloques desarrollados, así como la funcionalidad del mismo con dos casos de uso: el caso multicapa y el caso multidominio.

5.1. Definición del módulo de topología

El módulo de topología tiene dos funciones principales: (1) importar la información de estado de la red y (2) exportarla a elementos que la puedan consumir como son el PCE, el VNTM o el servidor ALTO. Existen varias formas para actualizar este módulo de topología, mediante protocolos de enrutamiento como el Border Gateway Protocol y su extensión Link State (BGP-LS), mediante OSPF o por medio de algún elemento de la arquitectura ABNO. Para este último caso se ha implementado una forma de comunicación a través de *web service*, que facilita la comunicación con entidades de alto nivel.

En la Fig. 24 se muestra el esquema del módulo de topología que se ha implementado. Como se puede observar, es el módulo que tiene almacenada la base de datos de la topología de red (Topology Database) en una base de datos única. Esta base de datos contiene distintas vistas o grafos, en este caso se muestra la capa IP, la de transporte y la de interconexión. También puede almacenar los parámetros TE, como el ancho de banda, longitud de onda de los enlaces, etc, que son necesarias para otros módulos de la arquitectura ABNO.

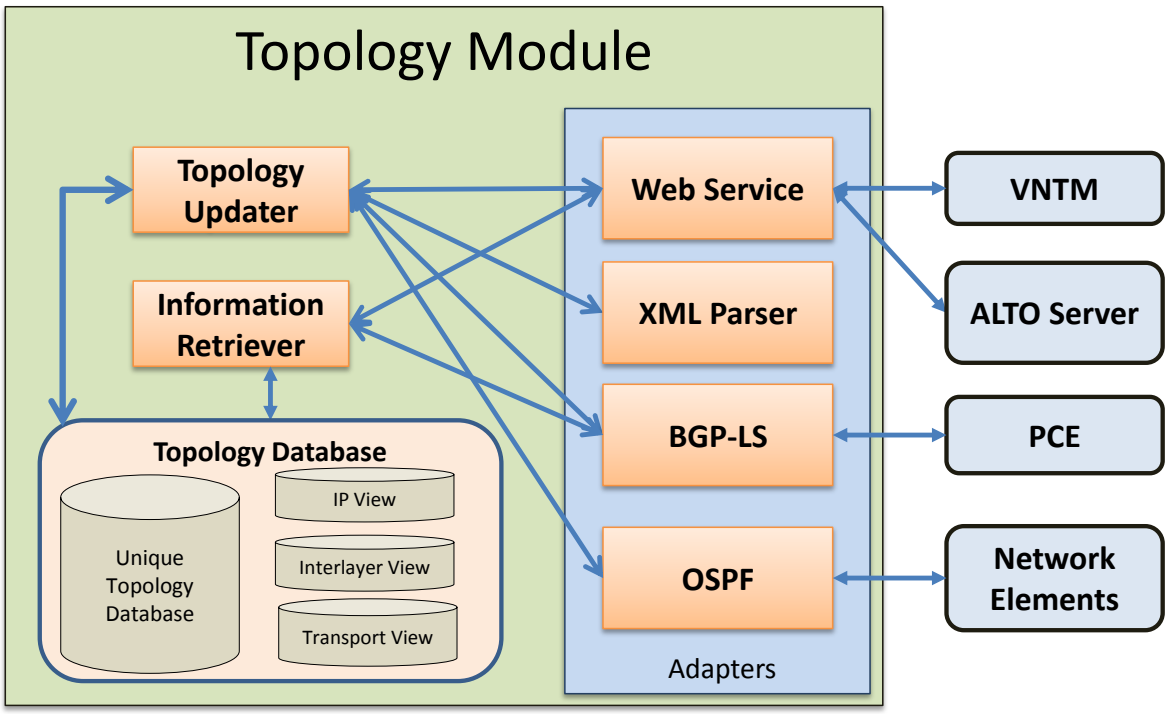

Fig. 24 Módulo de topología.

El módulo de topología tiene definidas varias interfaces para comunicarse con el resto de módulos de la arquitectura ABNO. Estos módulos pueden tanto actualizar la topología como pedir información acerca de ella. Para ello se han implementado los siguientes bloques:

- *Topology Updater*. Este bloque es la parte del módulo dedicada a mantener actualizada la topología con los recursos disponibles. Esta información la puede obtener de los adaptadores.
- *Information Retriever*. Este bloque se dedica de proveer la información requerida por los adaptadores.
- *Adaptadores*. Los adaptadores son elementos que reciben o envían información de otros módulos de la arquitectura ABNO o de fuentes externas como por ejemplo, los protocolos de enrutamiento. En este trabajo se han implementado cuatro módulos: *web service*, XML parser, OSPF y BGP-LS.

El primer adaptador que se define en la arquitectura es el *web service*. La interfaz desde el adaptador de *web service* al bloque "topology updater" e "information retriever" es la misma que se exporta a las entidades externas como el VNTM, PCE o ALTO. La diferencia es que las llamadas internas se realizan directamente con llamadas a las funciones, mientras que con las entidades externas se hace a través de http. A continuación se muestran las funciones relacionadas con el bloque "topology updater" y el bloque "information retriever".

- Topology Updater.
    - UpdateLink (link): inserta o actualiza un enlace entre dos nodos de la topología.
    - UpdateNode (node): inserta o actualiza un nodo en la topología.
    - UpdateIntf (interface): inserta o actualiza una interfaz perteneciente a un nodo de la topología.
    - DeleteLink (link): elimina un enlace entre dos nodos de la topología.
    - DeleteNode (node): elimina un nodo de la topología.
    - DeleteIntf (interface): elimina una interfaz perteneciente a un nodo de la topología.
- Information Retriever.
    - GetFullTopology (layer, domain ID): devuelve la información de la topología de una capa en concreto. En el caso en que la topología se componga de varios dominios, se puede especificar el dominio requerido.
    - GetOppositeNode(interface): devuelve el nodo de la capa opuesta al que la interfaz está conectada.
    - GetNeighbourNodesOf (node): devuelve los nodos vecinos, en la misma capa, de un nodo en concreto.
    - GetOppositeInterface(interface): devuelve la interfaz de la capa opuesta a la cual la interfaz está conectada.
    - GetNodeByName(node name): devuelve el nodo especificado por el nombre.
    - GetIntfByName(interface name): devuelve la interfaz especificada por el nombre.

Mientras que el adaptador de *web service* lo que hace es importar y exportar información desde el módulo de topología, tanto el adaptador "XML parser" como OSPF únicamente importan información en el módulo de topología. El adaptador "XML parser" permite leer de un fichero XML la topología y cargarla en la base de datos. Este adaptador es útil para iniciar en módulo con la información más estática. En entornos con plano de control, el módulo de topología puede escuchar directamente los protocolos de encaminamiento de plano de control. Para realizar este adaptador de OSPF se ha implementado un decodificador de OSPF que llama al bloque "topology updater" con la información de los enlaces y nodos que llega desde la red.

Finalmente, el adaptador de BGP-LS permite tanto importar como exportar la topología a módulos externos. Debido a que este protocolo es reciente en el estado del arte, se ha preferido explicarlo en más detalle en la siguiente sección.

5.2. BPG-LS

El protocolo Border Gateway Protocol-Link State (BGP-LS), definido en [23], es uno de los protocolos de enrutamiento que se puede usar para actualizar la topología. Este protocolo es una extensión del Border Gateway Protocol (BGP) [24] en el que se incluye la información del estado del enlace y de ingeniería de tráfico en el mensaje de actualización.

El protocolo BGP es un protocolo de enrutamiento entre sistemas autónomos (ASes), es decir, su función principal es intercambiar información de encaminamiento entre los distintos ASes. Esta incluye una lista de ASes accesibles y permite construir un grafo con la conexión entre ellos. Un sistema autónomo es un conjunto de routers que están bajo una única administración técnica, que usan uno o varios IGP y que tienen definido un conjunto de métricas comunes para determinar cómo encaminar los paquetes dentro del sistema autónomo. El protocolo BGP usa como protocolo de transporte el TCP [25], escuchando en el puerto TCP 179.

BGP-LS extiende los mensajes BGP Update (Fig. 25), para advertir de la información de estado de los enlaces de la topología. Esta información se envía en dos atributos del mensaje BGP Update, el MP_REACH (definido en [26]) y el LINK_STATE (definido en [23]).

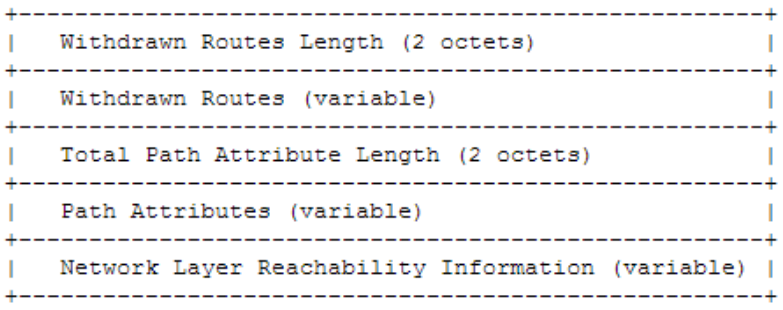

Fig. 25 Mensaje de actualización de BGP [24].

Para describir tanto los enlaces intra-dominio como los inter-dominio, en el atributo MP_REACH se incluye un campo llamado Link Network Layer Reachability Information (Link NLRI) (Fig. 26), que contiene los campos:

- *Local/Remote Node Descriptors*: que llevan la dirección origen y la de destino del enlace, respectivamente.
- *Link Descriptors* contiene una TLV (*Link Local/Remote Identifiers*), que transporta el prefijo de la interfaz no numerada (Unnumbered Interface).

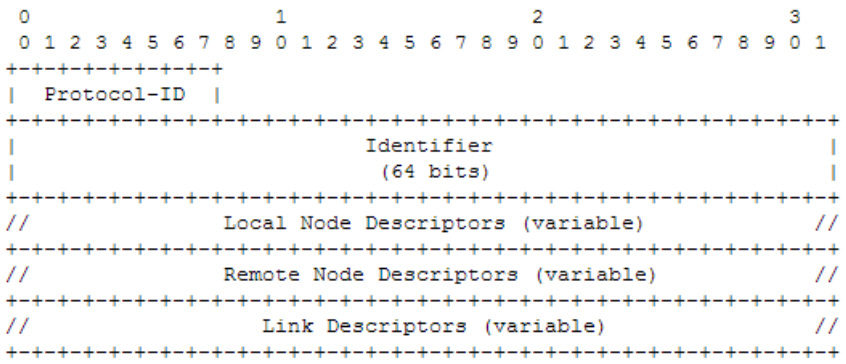

Fig. 26 Formato del Link NLRI [23]

La Fig. 27 muestra el mensaje de actualización extendido, el BGP-LS Update.

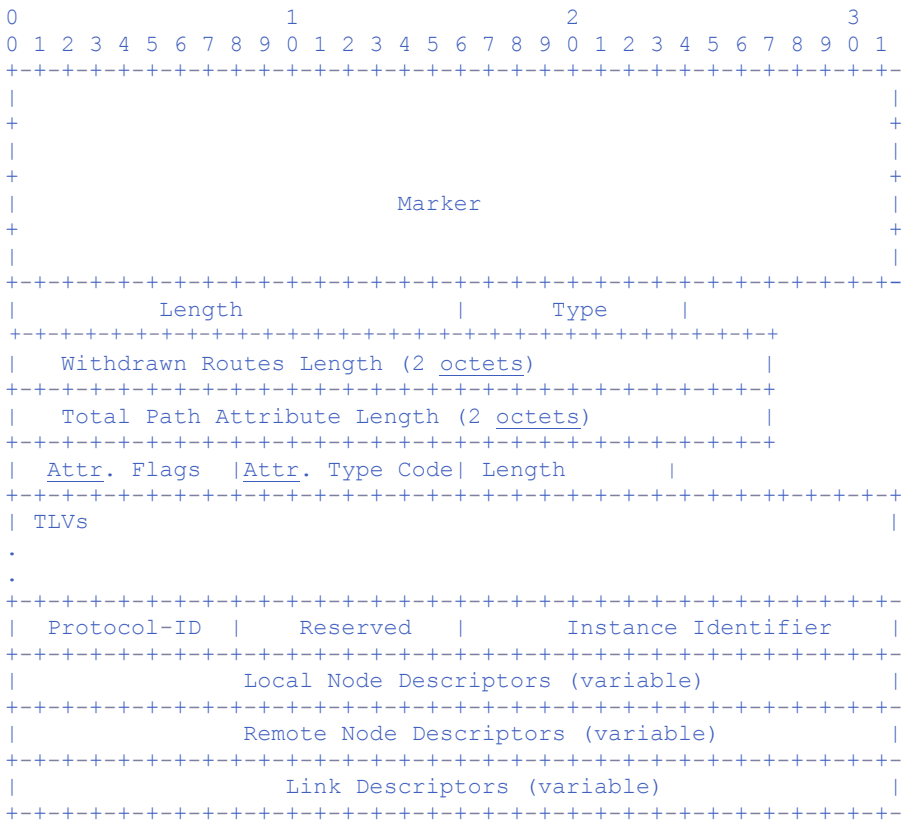

Fig. 27 Mensaje de actualización de BGP-LS.

En caso de que el mensaje informe sobre un enlace intra-dominio, la información de ingeniería de tráfico se incluye en el atributo LINK_STATE. Además, se puede añadir en ese atributo el TLV "Available Labels" [27] que incluye la información de las ranuras de frecuencia disponibles en el enlace.

El módulo de topología permite convertir un mensaje de OSPF a un mensaje de BGP-LS. Esto permite la lectura de mensajes OSPF de la red y posterior envío de mensajes BGP-LS a los BGP Speakers con la información de actualización indicada en el mensaje OSPF. La Fig. 28 muestra el mensaje de actualización de OSPF usado en nuestra implementación. Está compuesto por los siguientes campos:

- *Version*: versión del mensaje de OSPF. Se ha usado la versión 2.
- *Packet Lenght*: longitud del mensaje
- *Router ID*: dirección del nodo origen.
- *Area ID*: identifica el área donde se encuentra el router.
- *Checksum*: checksum de 16 bits en complemento a uno del paquete entero de OSPF.
- *AuType y Authentication*: cada paquete de OSPF se autentica. Los tipos de autenticación se asignan por el protocolo.
    - *LSAs*: este campo se denomina Link State Advertisements. Este es el campo que transporta la información TE del enlace. El mensaje OSPF Update puede tener varios LSAs.
        - La cabecera del LSA contiene los siguientes campos: *LS age, Options, LS type* (en nuestra implementación se usa el tipo 10), *Link State ID* (en nuestra implementación es 1 que corresponde con routers-LSA), *Advertising router, LS sequence number, LS checksum, lenght*.
        - El cuerpo del LSA es el formato del router-LSA, que se compone de los campos: *Link Type, Link ID* (que transporta la dirección destino del enlace) y *TLVs* (donde se incluye la información TE).

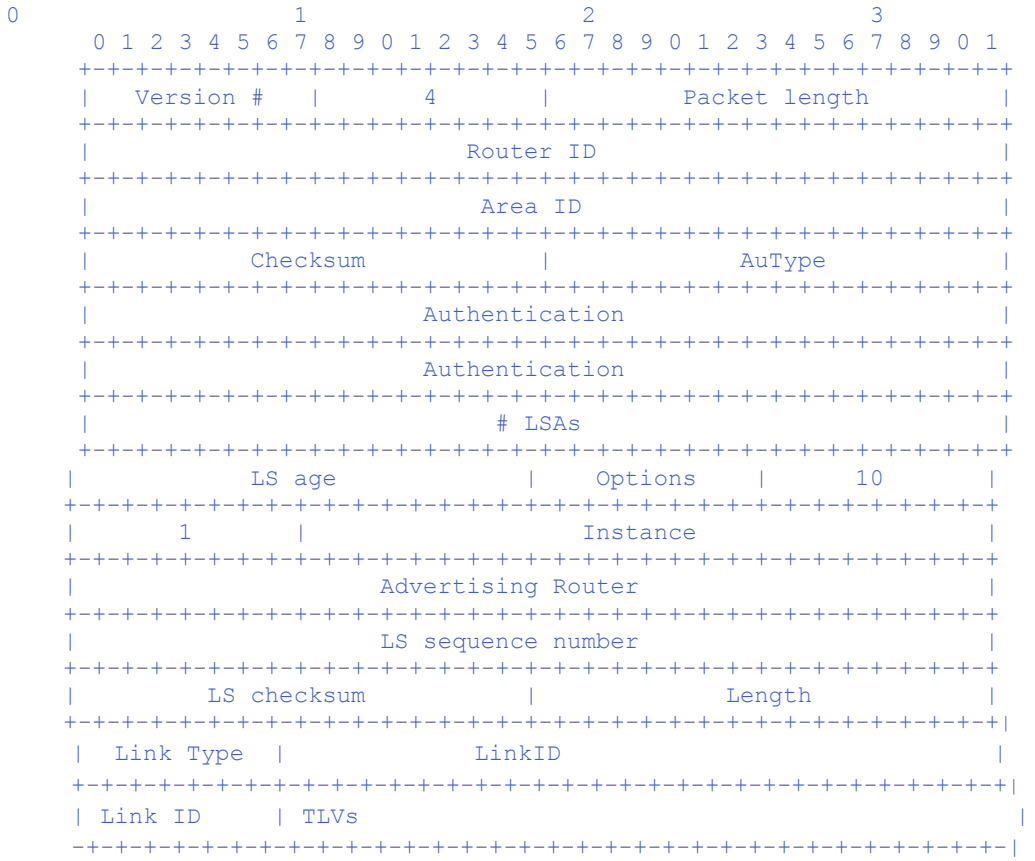

Fig. 28 Formato de mensaje de actualización de OSPF (OSPF Update).

La conversión del mensaje OSPF a BGP-LS se muestra en la siguiente tabla:

OFPS	**BGP-LS**
Router ID	Link Node Descriptor

Link ID	Remote Node Descriptor
TLVs (Link Local/Remote Identifiers)	Link Descriptors
TLVs (maximum bandwidth, maximum reservable bandwidth, unreserved bandwidth, available labels, …)	TLVs

Tabla 2 Conversión de mensaje OSPF en BGP-LS.

5.3. Casos de uso multicapa

5.3.1. Definición

El siguiente caso de uso describe cómo la arquitectura ABNO puede proveer un enlace IP entre dos routers automáticamente y sin necesidad de intervención manual. Existen distintas formas de interacción entre los módulos para realizar esta provisión, aquí se ha asumido que no hay ABNO controller ni Provisioning Manager.

La red IP/MPLS está desplegada sobre una red de transporte óptico, que se usa para dar conectividad a la red IP. La necesidad de reservar un enlace IP es una operación básica de los operadores de red. Esta operación se usa para ofrecer nuevos servicios a los clientes, como la conectividad Internet, VPN, IPTV, etc. Cuando los operadores despliegan nuevas infraestructuras es necesario desplegar nuevo equipamiento IP. Este proceso requiere intervención manual y suele ser programado y desplegado periódicamente. Normalmente la confidencialidad no es un tema importante ya que las dos capas de red (la capa óptica o de transporte y la capa IP) son administradas y gestionadas por distintos departamentos dentro del mismo operador.

Una vez el equipo está instalado en la red y los operadores reciben peticiones de crear nuevos enlaces IP entre dos nodos, el departamento IP identifica una interfaz libre para cada nodo. Este departamento tiene que pedir al departamento de transporte que cree un *lightpath* entre estos dos nodos. Si es posible el establecimiento de un enlace óptico entre esos nodos, el departamento de transporte configura el equipamiento óptico y establece la conexión. Cuando el departamento IP recibe el *lightpath* establecido, los routers IP se configuran conforme al tipo de servicio. Este proceso de configuración puede llevar varios días, incluso cuando los elementos de red ya están activos y establecidos en la red.

Con la arquitectura ABNO se puede automatizar este proceso de configuración utilizando las tecnologías del plano de control y usando una interfaz para configurar los routers IP. También se pretende que sea capaz de tratar las peticiones que los operadores realizan para establecer nuevas conexiones conteniendo parámetros TE como el ancho de banda, el retardo máximo o el jitter. Usando el estado de la red, ABNO computa un camino en la capa de transporte conforme a las restricciones requeridas. Para este propósito, puede interactuar con uno o varios PCEs. Las propuestas existentes para el cómputo de caminos multicapa pueden categorizarse en dos grupos:

- Un único PCE puede computar el camino multicapa de principio a fin. En este caso almacenaría en su TED la información de ambas capas de red.
- Varios PCEs, cada uno encargado de una capa, computan el camino.

Una de las ventajas de la arquitectura ABNO es que no es necesario usar todos los módulos a la vez siempre. Como ejemplo en este caso de uso (Fig. 29) se utilizan módulos de la arquitectura ABNO como el PCE y el VNTM, pero no otros como el ABNO controller o el Provisioning Manager. La Fig. 29 muestra el cálculo de un camino multicapa en una red IP/MPLS sobre WSON. Tiene 4 componentes principales:

- El cliente (nodo R1).
- El PCE encargado de la capa IP/MPLS.
- El PCE encargado de la capa WSON.
- VNTM.

El proceso de cálculo comienza cuando el cliente pide un camino al PCE IP/MPLS (1), donde el cliente puede ser un MPLS Label Switch Router (LSR) como el R1, o puede ser un NMS. Una vez el PCE recibe la petición, busca si en la capa IP/MPLS hay recursos suficientes y en caso de que hubiera, manda la respuesta con el camino al cliente. Si por el contrario, los recursos son insuficientes, el PCE de la capa IP/MPLS realiza una petición donde le pide al PCE de la capa WSON que reserve recursos de dicha capa para conectar los routers IP (2). Normalmente el PCE IP/MPLS realiza más de una petición al PCE WSON, dado que no tiene la topología de la capa óptica. Una vez recibe la respuesta del PCE de la capa WSON, el IP/MPLS PCE elige el circuito que más se ajuste a sus necesidades, lo encapsula en un objeto ERO que forma parte de la respuesta y la manda al cliente (3). En caso de que se haya computado un camino multicapa, el cliente envía un mensaje con el objeto ERO al VNTM (4), que se encarga de iniciar la instalación del circuito en la capa WSON (5). Finalmente, el cliente inicia la reserva de recursos en la capa IP/MPLS (6).

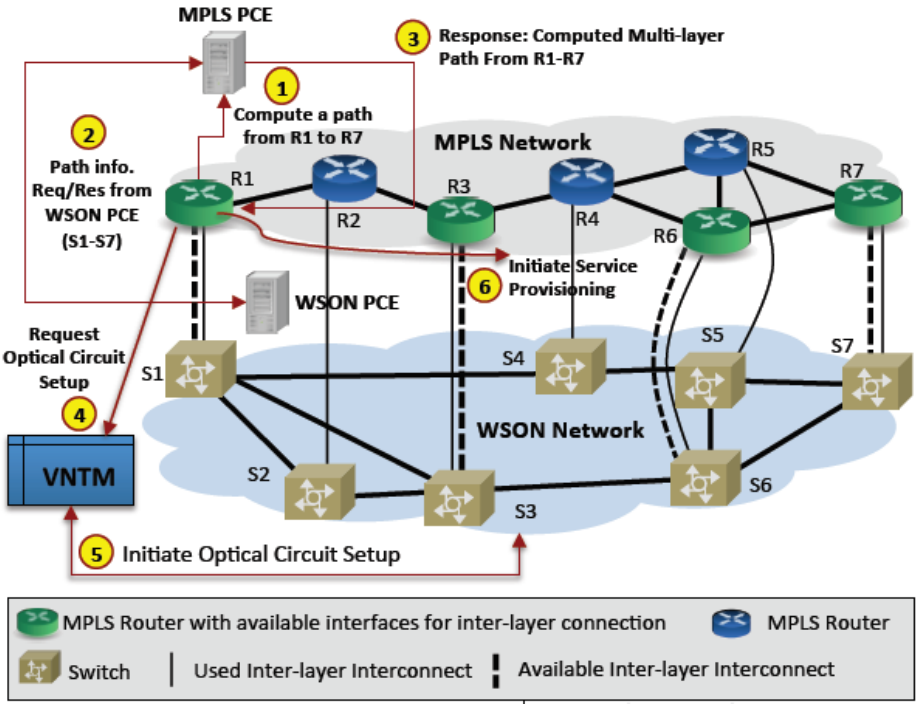

Fig. 29 Construcción de un camino multicapa para una red IP/MPLS sobre WSON [14].

5.3.2. Validación

Para validar este caso de uso se ha implementado un módulo de topología siguiendo el formato mostrado en Fig. 24. Este módulo almacena la topología en una base de datos interna que puede ser introducida y actualizada leyendo un fichero XML, escuchando mensajes OSPF de la red o por mensajes de actualización enviados por otros módulos de la arquitectura. La comunicación con estos módulos se hace a través de *web service*. Las funciones implementadas son las descritas en la sección 5.1 y permiten actualizar la topología y devolver su información.

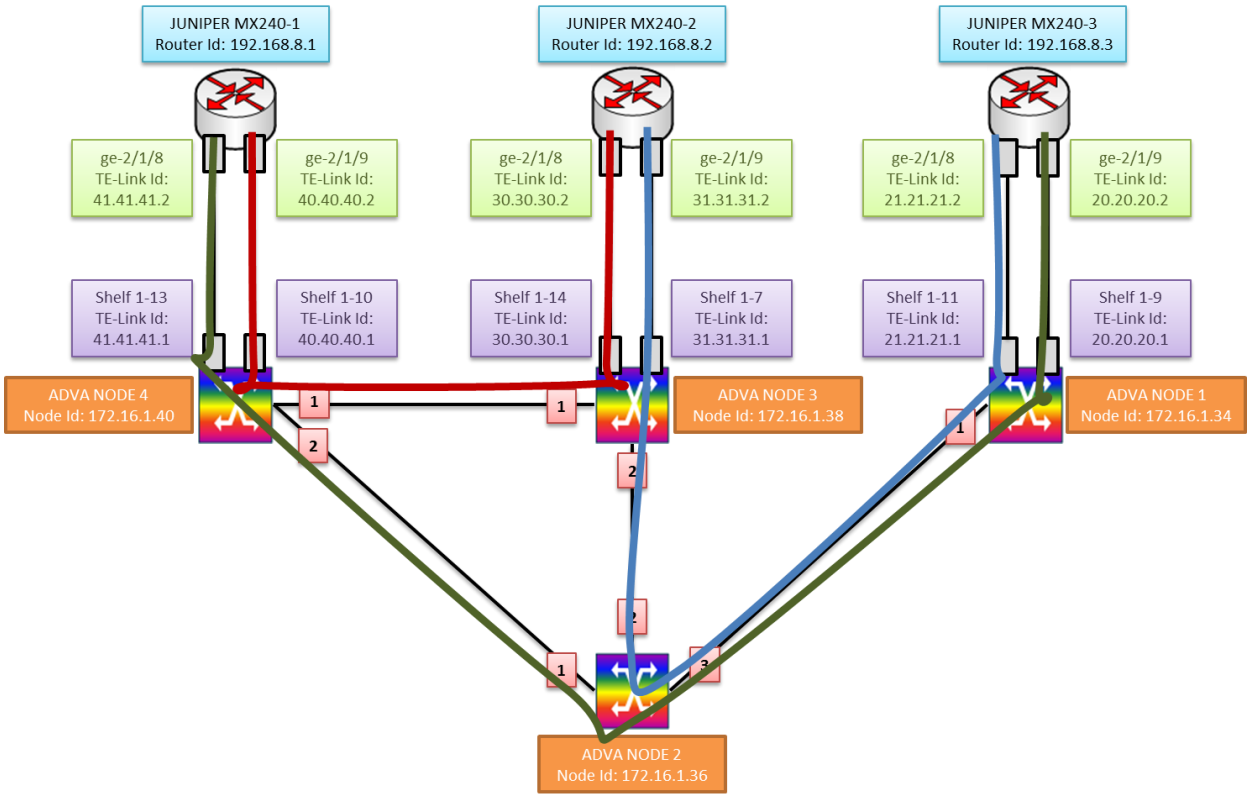

Fig. 30 Topología usada para validar el caso multicapa.

Este módulo se ha validado en una red multicapa mostrada en la Fig. 30. Este escenario se compone de cuatro ROADMs en la capa óptica y tres IP/MPLS routers. En la experimentación el PCE periódicamente obtiene la topología de la capa óptica. Para ello realiza la pregunta "GetFullTopology (transport, 1)" al "information updater" a través del *web service*. Los parámetros de la petición se encapsulan en un mensaje JSON. El siguiente ejemplo muestra la petición realizada por el PCE y la respuesta obtenida del módulo de topología:

Petición:

```
{"layer":"transport", "domainID":"1"}
```

Respuesta:

```
{"nodeID":"ADVA_NODE_4",
"address":["172.16.1.40"],
"isPhysical":true,
"intfList":[
       {"name":"ADVA_NODE_4_Shelf1-13",
"address":["41.41.41.1"],"layering":["transport"],"isPhysical":true,"intfUp":true},
       {"name":"ADVA_NODE_4_Shelf1-10",
"address":["40.40.40.1"],"layering":["transport"],"isPhysical":true,"intfUp":true},
       {"name":"ADVA_NODE_4_Unnumbered1","address":["1"],"layering":["transport"],"isPhysica
l":true,"intfUp":true},
       {"name":"ADVA_NODE_4_Unnumbered2","address":["2"],"layering":["transport"],"isPhysica
l":true,"intfUp":true}
       ],
"domain":1,
"layer":"transport"}
{"nodeID":"ADVA_NODE_2",
"address":["172.16.1.36"],
"isPhysical":true,
"intfList":[
```

```
      {"name":"ADVA_NODE_2_Unnumbered1","address":["1"],"layering":["transport"],"isPhysica
l":true,"intfUp":true},
      {"name":"ADVA_NODE_2_Unnumbered2","address":["2"],"layering":["transport"],"isPhysica
l":true,"intfUp":true},
      {"name":"ADVA_NODE_2_Unnumbered3","address":["3"],"layering":["transport"],"isPhysica
l":true,"intfUp":true}
      ],
"domain":1,
"layer":"transport"}
{"nodeID":"ADVA_NODE_3",
"address":["172.16.1.38"],
"isPhysical":true,
"intfList":[
      {"name":"ADVA_NODE_3_Shelf1-
7","address":["31.31.31.1"],"layering":["transport"],"isPhysical":true,"intfUp":true},
      {"name":"ADVA_NODE_3_Shelf1-
14","address":["30.30.30.1"],"layering":["transport"],"isPhysical":true,"intfUp":true},
      {"name":"ADVA_NODE_3_Unnumbered1","address":["1"],"layering":["transport"],"isPhysica
l":true,"intfUp":true},
      {"name":"ADVA_NODE_3_Unnumbered2","address":["2"],"layering":["transport"],"isPhysica
l":true,"intfUp":true}
      ],
"domain":1,
"layer":"transport"}
{"nodeID":"ADVA_NODE_1",
"address":["172.16.1.34"],
"isPhysical":true,
"intfList":[
      {"name":"ADVA_NODE_1_Shelf1-
11","address":["21.21.21.1"],"layering":["transport"],"isPhysical":true,"intfUp":true},
      {"name":"ADVA_NODE_1_Shelf1-
9","address":["20.20.20.1"],"layering":["transport"],"isPhysical":true,"intfUp":true},
      {"name":"ADVA_NODE_1_Unnumbered1","address":["1"],"layering":["transport"],"isPhysica
l":true,"intfUp":true}
      ],
"domain":1,
"layer":"transport"}
{"linkID":"ADVA_NODE_4_ADVA_NODE_3","isDirectional":true,
"source":{"node":"ADVA_NODE_4","intf":"ADVA_NODE_4_Unnumbered1"},
"dest":{"node":"ADVA_NODE_3","intf":"ADVA_NODE_3_Unnumbered1"},
"type":"intradomain","teMetric":0.0}
{"linkID":"ADVA_NODE_4_ADVA_NODE_2","isDirectional":true,
"source":{"node":"ADVA_NODE_4","intf":"ADVA_NODE_4_Unnumbered2"},
"dest":{"node":"ADVA_NODE_2","intf":"ADVA_NODE_2_Unnumbered1"},
"type":"intradomain","teMetric":0.0}
{"linkID":"ADVA_NODE_3_ADVA_NODE_2","isDirectional":true,
"source":{"node":"ADVA_NODE_3","intf":"ADVA_NODE_3_Unnumbered2"},
"dest":{"node":"ADVA_NODE_2","intf":"ADVA_NODE_2_Unnumbered2"},
"type":"intradomain","teMetric":0.0}
{"linkID":"ADVA_NODE_2_ADVA_NODE_1","isDirectional":true,
"source":{"node":"ADVA_NODE_2","intf":"ADVA_NODE_2_Unnumbered3"},
"dest":{"node":"ADVA_NODE_1","intf":"ADVA_NODE_1_Unnumbered1"},
"type":"intradomain","teMetric":0.0}
```


En caso de que se pida que devuelva un nodo especificando el nombre, el VNTM preguntaría usando la función GetNodeByName(MX240-3) del *web service*. A continuación se muestra el ejemplo:

Petición:


```
{"nodeName":"MX240.3"}
```

Respuesta:


```
{"nodeID":"MX240-3","address":["192.168.8.3"],"isPhysical":true,
"intfList":
       [{"name":"MX240-3_ge-2/1/8",
       "address":["21.21.21.2"],"layering":["IP"],"isPhysical":true,"intfUp":true},
       {"name":"MX240-3_ge-2/1/9",
       "address":["20.20.20.2"],"layering":["IP"],"isPhysical":true,"intfUp":true}],
"domain":1,"layer":"IP"}
```


Estas funciones se explican en detalle en el ANEXO A.

5.4. Casos de uso multidominio

5.4.1. Definición

Una red está compuesta por múltiples dominios que pueden pertenecer a distintos operadores y estar compuestos por distinta tecnología. Por lo tanto, el cálculo de una ruta que tenga que pasar varios dominios puede resultar complicado.

Una arquitectura que permite el cálculo de estas rutas es la del PCE Jerárquico (Hierarchical PCE - H-PCE). Un ejemplo se puede ver en la Fig. 31. En esta arquitectura existen varios PCEs organizados por niveles. Existe un PCE, llamado PCE hijo, encargado de cada dominio, que es el de nivel más bajo ya que solo tendría conocimiento de la topología de red de su dominio. Por encima, se encuentra el PCE padre (H-PCE) que tiene constancia de todos los dominios y de los PCEs encargados de cada uno. El H-PCE puede tener conocimiento de toda la topología de red, incluidos los enlaces intra-dominio, o no tener conocimiento de toda la red al completo sino que sólo de los enlaces que existen entre los dominios (inter-dominio). En este último caso, el cálculo de una ruta es más complejo: la petición es dirigida al PCE encargado del dominio (el PCE1 en la Fig. 31). Éste comprueba si el nodo de destino pertenece a su dominio. En caso afirmativo, calcula la ruta y la devuelve sin implicar a ningún PCE. En caso contrario, redirige la petición al H-PCE, que se encargará de comprobar el dominio al que pertenece el nodo destino y los dominios por los que será necesario establecer la ruta. El H-PCE se comunicará con los PCEs de cada dominio implicado, solicitando una sub-ruta por ese dominio. En el ejemplo de la Fig. 31, el H-PCE se comunica con los dominios 2, 3 y 4. Al PCE2 le pide una ruta de los Nodos 2-1 A y B a los Nodos 2-3 A y B, al PCE4 le pide una ruta de los Nodos 4-1 A y B a los Nodos 4-3 A y B y por último, al PCE3 le pide la ruta de los Nodos 3-2 A y B al nodo destino. Una vez reunidas todas las respuestas, el H-PCE se encarga de combinarlas para devolver el camino más adecuado.

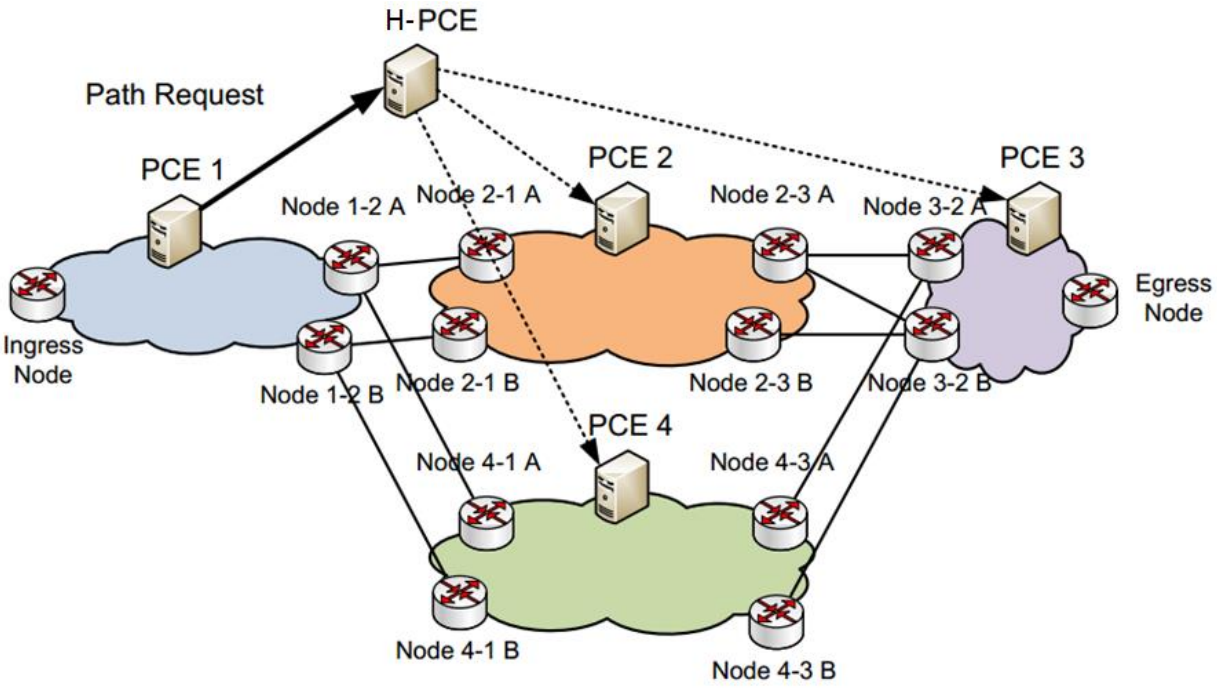

Fig. 31 Cálculo de una ruta multidominio en un entorno de H-PCE [15].

Esta arquitectura necesita obtener la topología de la red para poder realizar el cálculo de las rutas. Por ese motivo se dota en este trabajo al módulo de topología de soporte para BGP-LS. La arquitectura H-PCE con BGP-LS se muestra en la Fig. 32. Cada dominio tiene un PCE, llamado PCE hijo (cPCE), capaz de computar los caminos dentro de su dominio. Este cPCE almacena la topología de su dominio en su TED local, llamada TED de dominio, que se ha construido usando la información de IGP. En cada dominio, existe un BGP Speaker que tiene acceso a dicha TED de dominio y la envía al PCE padre (pPCE). En el lado del PCE padre hay un BGP Speaker que mantiene la sesión BGP con cada uno de los BGP Speakers de los dominios para recibir sus respectivas topologías y construir con ello la topología completa de la red y almacenarla en la TED padre. La información de la topología que los hijos envían al padre se rige por una política específica, ya que podrían mandar la información de toda la topología interior del dominio o solo mandar la información de los enlaces entre los dominios.

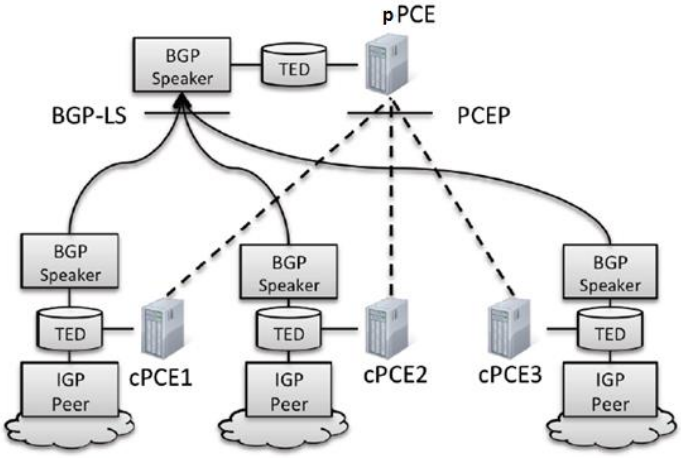

Fig. 32 Arquitectura H-PCE usando el protocolo BGP-LS [28].

La mínima información que se puede enviar es la conectividad inter-dominio, incluyendo la información de TE de los enlaces inter-dominio [29]. Con solo esta información, el pPCE es capaz de calcular una ruta que pase por cualquiera de los dominios con ayuda de los PCEs hijos, ya que tendrá conectados todos los dominios (Distributed Multi-Domain Minimum Transit). Otra posibilidad es que el BGP Speaker de los hijos esté configurado para enviar una lista completa de todos los enlaces TE internos del dominio. En este caso, el PCE padre tendrá acceso a una base de datos extendida formada por todos los dominios, con visibilidad de enlaces intra-dominio e inter-dominio (Local Multi-Domain Minimum Transit). Estos dos algoritmos para el cálculo de los caminos se han usado en este trabajo:

- **Distributed Multi-Domain Minimum Transit Domains (DMDMTD).** En este algoritmo la topología de red está distribuida, cada cPCE almacena la topología de su dominio junto con los enlaces inter-dominio que la conectan con otros, y el pPCE almacena la topología de enlaces inter-dominio que conectan todos los dominios de la red.
  La Fig. 33 muestra un flujo de mensajes PCEP para la petición de un camino multidominio en el caso de usar el algoritmo DMDMTD. El proceso comenzaría con la petición desde el nodo destino de transporte al cPCE de su dominio, en el ejemplo sería el Dominio 1. El cPCE del dominio 1 comprueba si el nodo destino se encuentra dentro de su dominio, como no es el caso, redirige la petición al pPCE. Éste comprueba el dominio donde se encuentra el nodo destino, y calcula los dominios por los que debe pasar la respuesta. Envía peticiones de cálculo de camino a aquellos dominios implicados (dominio 2 y dominio 3) y con las respuestas enviadas por estos, selecciona el camino óptimo de acuerdo al algoritmo y políticas que implemente. Finalmente, devuelve la ruta completa al cPCE del dominio 1.
- **Local Multi-Domain Minimum Transit (LMDMT).** En este algoritmo el pPCE almacena en la TED toda la topología de red, tanto los enlaces inter-dominio que conectan los dominios como los intra-dominio que conectan los nodos de cada uno. El cálculo de una ruta para este algoritmo es más sencillo, ya que el H-PCE podrá computar cualquier ruta de principio a fin sin necesidad de los PCEs hijos y con mejor precisión.

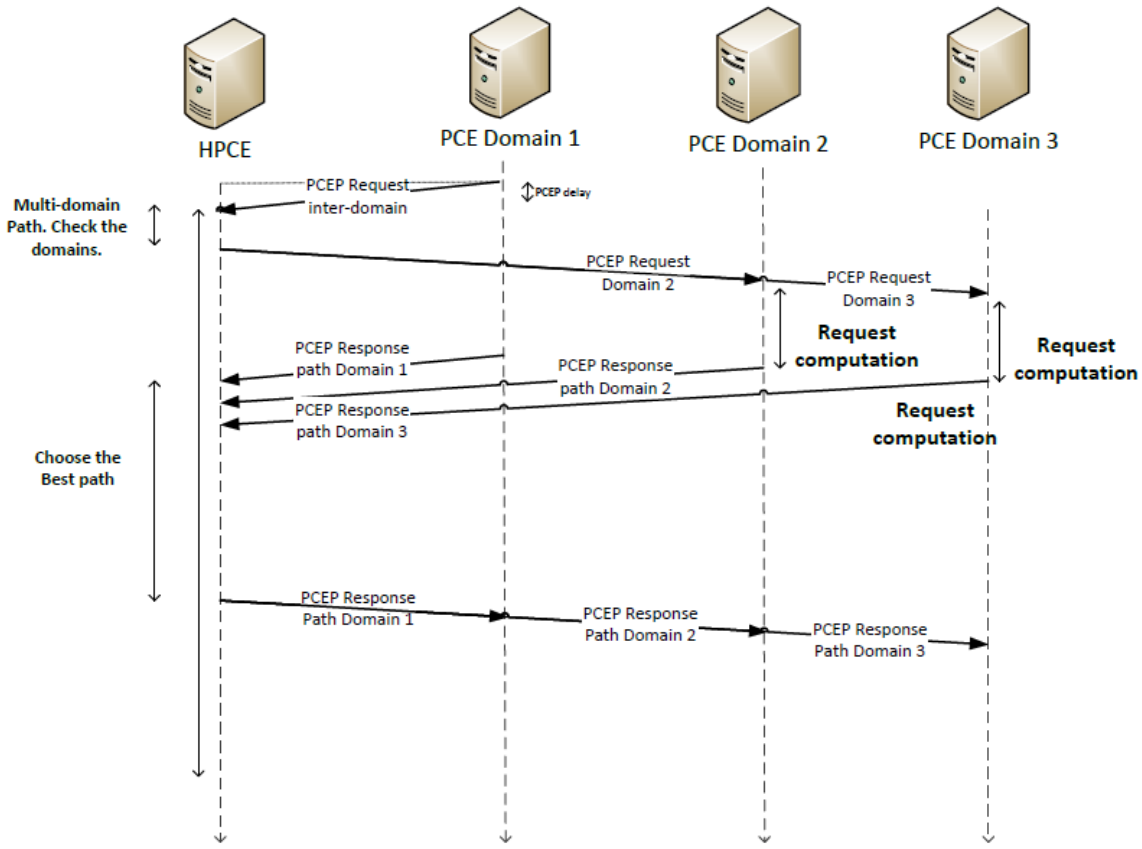

Fig. 33 Flujo de mensajes para el cálculo de un camino multidominio con el algoritmo DMDMTD

5.4.2. Validación

Se ha implementado la arquitectura de H-PCE con BGP-LS para redes elásticas, tal y como se ha explicado anteriormente. El escenario usado para la validación es una red óptica elástica multidominio. Este escenario se compone por tres dominios, cada uno con 7 nodos y un cPCE. Todos los nodos son máquinas virtuales en un servidor con dos procesadores Intel Xeon E5-2630 2.30 GHz (con 6 núcleos cada uno) y 192 GB de memoria RAM. El retardo entre los nodos y su cPCE respectivo y entre el pPCE y los cPCE es de 25 ms. Este retardo se ha puesto usando la herramienta netem.

Se puede ver un ejemplo en la captura de Wireshark de la Fig. 34. Se puede ver el intercambio de mensajes BGP-LS para abrir una sesión (OPEN Message y KEEPALIVE Message) y para actualizar la topología (UPDATE Message).

12	26.203094	192.168.1.200	192.168.1.201	BGP	OPEN Message
14	26.215737	192.168.1.201	192.168.1.200	BGP	OPEN Message
20	26.260332	192.168.1.201	192.168.1.200	BGP	KEEPALIVE Message
21	26.264338	192.168.1.200	192.168.1.201	BGP	KEEPALIVE Message
164	62.572986	192.168.2.200	192.168.1.201	BGP	OPEN Message
166	62.596196	192.168.1.201	192.168.2.200	BGP	OPEN Message
168	62.600891	192.168.1.201	192.168.2.200	BGP	KEEPALIVE Message
174	62.632379	192.168.2.200	192.168.1.201	BGP	KEEPALIVE Message
464	101.332796	192.168.3.200	192.168.1.201	BGP	OPEN Message
465	101.346238	192.168.1.201	192.168.3.200	BGP	OPEN Message
469	101.361512	192.168.1.201	192.168.3.200	BGP	KEEPALIVE Message
472	101.383922	192.168.3.200	192.168.1.201	BGP	KEEPALIVE Message
484	102.533076	192.168.2.200	192.168.1.201	BGP	UPDATE Message
485	102.543051	192.168.2.200	192.168.1.201	BGP	UPDATE Message

Fig. 34 Intercambio de mensajes BGP-LS [28].

La Fig. 35 muestra el intercambio de mensajes para realizar y computar una petición multidominio en una arquitectura de H-PCE con dos algoritmos que han sido implementados. En el LMDMTD, la política BGP está configurada para mandar toda la topología de los dominios, y el pPCE calcula las rutas directamente, basándose en la secuencia más corta de dominios por los que dirigir la petición, y luego, dentro de cada dominio aplica el algoritmo RSA (Fig. 35 arriba a la izquierda). Si la petición es dentro del mismo dominio, los cPCEs usan el mismo algoritmo RSA. El otro algoritmo DMDMTD, los BGP Speakers de los hijos solo mandan los enlaces inter-dominio al padre. Por lo tanto, a la hora de calcular un camino, se usa el procedimiento clásico de la arquitectura H-PCE donde el pPCE pregunta a los cPCEs por el camino en sus respectivos dominios (Fig. 35 abajo a la izquierda). En la misma imagen a la derecha se muestra el ERO enviado por el pPCE, con el objeto Label Control donde se muestra la ranura de frecuencia que se reservará para ese camino.

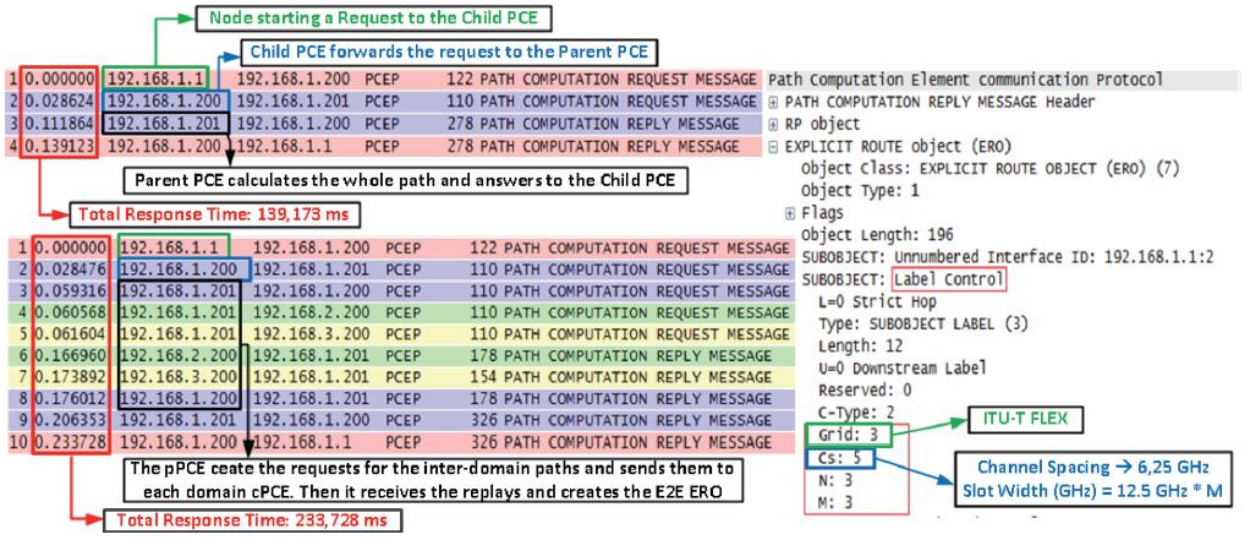

Fig. 35 Flujo de mensajes PCEP para una petición multidominio con LMDMTD (arriba a la izquierda) y con DMDMTD (abajo a la izquierda). Mensaje ERO detallado (derecha) [28].

5.5. Conclusiones

El módulo de topología es el encargado de almacenar y proporcionar la topología de red a los demás módulos de la arquitectura ABNO. En esta sección se ha diseñado, implementado y validado este módulo. El diseño se compone de tres bloques: Information Retriever, encargado de proporcionar la información de la topología, Topology Updater, encargado de crearla y actualizarla y los Adaptadores, que se encargan de enviar o recibir la información de los demás módulos de la arquitectura.

Se ha validado en dos casos de uso, el multicapa y multidominio. En el primero se valida cómo la arquitectura ABNO puede proveer un enlace IP entre dos routers automáticamente, para ello, el módulo de topología almacena una red IP/MPLS sobre una red de transporte óptico, y los módulos de la arquitectura piden o actualizan a través de los adaptadores dicha topología. El segundo estudia cómo la arquitectura de H-PCE es capaz de proporcionar un camino en una multidominio. La arquitectura H-PCE se compone de varios PCEs hijos encargados de cada dominio y de un PCE padre que se encarga de calcular el camino multidominio ayudado por los hijos. Cada PCE se basa en el cálculo en su TED local, y para que ésta esté actualizada se dota al módulo de topología de soporte BGP-LS que envía y recibe mensajes de actualización de enlaces inter-dominio. Se evalúan dos algoritmos de cálculo, el DMDMTD, donde el PCE padre solo almacena en su TED los enlaces inter-dominio y el LMDMT, donde el PCE padre almacena también los enlaces intra-dominio de dominios.

6. Conclusiones y futuras líneas de trabajo

6.1. Conclusiones

En este trabajo final de máster se ha contribuido a dos módulos de la arquitectura ABNO: el PCE y el módulo de topología, el cuál además se ha implementado por completo.

La contribución al PCE está relacionada con la implementación y validación de mecanismos para evitar el robo de lambda. El proceso de actualización de la topología puede llevar un tiempo, por lo que el cálculo de las rutas puede basarse en información desactualizada con respecto al actual estado de la red. En estos casos, se produce bloqueo por robo de lambda. Para evitar este bloqueo se ha definido el mecanismo de Pre-Reserva que reserva los recursos de la red de los caminos calculados. Se ha evaluado este mecanismo de pre-reserva en dos escenarios:

- **Escenario de restauración masiva:** En la evaluación del mecanismo de PR para el escenario de restauración masiva se ha comprobado que dicho mecanismo permite eliminar el bloqueo de lambda al calcular las peticiones y por tanto, minimizar el tiempo de restablecimiento del tráfico en la red.
- **Escenario de múltiples:** PCEs: los PCEs colaborativos y los PCEs con división de recursos de la red ofrecen los mejores resultados en términos de bloqueo por robo de lambda cuando el PCE implementa el mecanismo de PR. Los PCEs colaborativos minimizan este bloqueo debido al intercambio de mensajes de notificación con la información de los recursos reservados. Los PCEs con división de recursos de la red actúan cada uno como un único PCE pero calculando las rutas sobre menos recursos ya que éstos no son compartidos.

La segunda contribución del trabajo es el desarrollo e implementación un módulo de topología para la arquitectura ABNO. El objetivo del módulo de topología es almacenar la topología de la red y mantenerla actualizada para proporcionar esta información a módulos de la arquitectura red. Este módulo se ha evaluado en dos escenarios, uno multicapa y otro multidominio con la arquitectura H-PCE. Se han implementado tres maneras de actualizar el módulo de topología, leyendo un fichero XML, por *web service* o leyendo y procesando mensajes de enrutamiento enviados por los nodos, concretamente los protocolos OSPF y BGP-LS. Desde el punto de vista de la exportación se ha utilizado un *web service* y BGP-LS para enviar la información a otros módulos.

6.2. Trabajo Futuro

Existen varias líneas de trabajo futuro. En el entorno de múltiples PCEs con división de los recursos de la red se podrían estudiar nuevos escenarios donde los PCCs demandan caminos a los PCEs con distintas tasas. En ese escenario, sería necesario que los recursos de la red fuesen asignados dinámicamente a los PCEs. Estos recursos no serían nunca compartidos, pero podría darse el caso de que si un PCE se queda sin recursos, pueda trabajar con los recursos de la red de otro PCE que estuviesen libres.

Otra línea de investigación sería la comparación de rendimiento en términos de bloqueo por robo de lambda con distintos algoritmos de cálculo de rutas en los escenarios de mútiples PCEs y de restauración masiva de la red.

Desde el punto de vista del módulo de topología el principal trabajo sería la definición de un interfaz estándar para poder compartir la información topológica de la red. Para ello sería necesario realizar primero un modelo de YANG común y después utilizarlo ya sea a través de una interfaz JSON o Netconf.

ANEXO A. Manual de usuario

A.1.1. Topology Updater

Las funciones del Topology Updater son las siguientes:

- **UpdateNode**: inserta un nodo en el grafo. Si el nodo no está insertado se inserta y si el nodo ya está en la red se reemplaza por este nuevo.
    - Parámetros de entrada:
        - http://URL/TopologyModule/UpdateNode
        - JSON String (<Key, Value>) conteniendo los siguientes parámetros:

Field	***Description***
NodeName	Nombre del nodo.
Address	Dirección del nodo.
IsPhysical (optional field)	Especifica si el nodo es físico o no. Es un booleano.
Domain	Dominio al que pertenece el nodo.
LayerNode	Capa donde se encuentra el nodo.
ParentRouter (optional field)	El router padre.
xLocation (optional field)	Coordenada x donde se encuentra el nodo situado.
yLocation (optional field)	Coordenada y donde se encuentra el nodo situado.

    - Salida:
        - Objeto JSON conteniendo un string: True or false

- **UpdateLink**: inserta un enlace entre dos nodos. Los nodos y las interfaces deben existir, si no se devuelve error.
    - Parámetros de entrada:
        - http://URL/TopologyModule/UpdateLink
        - JSON String (<Key, Value>) conteniendo los siguientes parámetros:

Field	***Description***
linkID	Identificador único del enlace.
IsDirectional (optional field)	Campo que especifica si el enlace es bidireccional o no.
srcNode	Nombre del nodo destino.
srcIntf	Interfaz del nodo origen a la que el enlace se conectará
dstNode	Nombre del nodo destino.
dstIntf	Interfaz del nodo destino a la que el enlace se conectará.
LayerLink	Capa a la que pertenece el enlace. Puede ser :'transport', 'IP' or 'interlayer'
TypeLink (optional field)	Campo indicando el tipo de enlace: intradomain, interdomain or interlayer
teMetric (optional field)	Métrica del enlace

    - Salida:
        - Objeto JSON conteniendo un string: True or false

- **UpdateIntf**: inserta una interfaz en un nodo. El nodo debe existir. Si la interfaz ya existe en ese nodo, se reemplaza por la nueva.
    - Parámetros de entrada:

- http://URL/TopologyModule/UpdateIntf
- JSON String (<Key, Value>) conteniendo los siguientes parámetros:

Field	***Description***
NodeName	Nombre del nodo.
IntfName	Nombre de la interfaz.
Address	Dirección de la interfaz.
Layering	Capa a la que pertenece la interfaz. Puede ser: 'transport' or 'IP'
IsPhysical (optional field)	Indica si la interfaz es física o no.
intfUp (optional field)	Indica si la interfaz está active o no.

- Salida:
  - Objeto JSON conteniendo un string: True or false

- **DeleteNode**: elimina el nodo del grafo.
  - Parámetros de entrada:
    - http://URL/TopologyModule/DeleteNode
    - JSON String (<Key, Value>) conteniendo los siguientes parámetros:

Field	***Description***
NodeName	Nombre único del nodo.

  - Salida:
    - Objeto JSON conteniendo un string: True or false

- **DeleteLink**: elimina el enlace del grafo.
  - Parámetros de entrada:
    - http://URL/TopologyModule/DeleteLink
    - JSON String (<Key, Value>) conteniendo los siguientes parámetros:

Field	***Description***
linkID	Identificador único del enlace.

  - Salida:
    - Objeto JSON conteniendo un string: True or false

- **DeleteIntf**: elimina el enlace del grafo.
  - Parámetros de entrada:
    - http://URL/TopologyModule/DeleteIntf
    - JSON String (<Key, Value>) conteniendo los siguientes parámetros:

Field	***Description***
NodeName	Nombre del nodo
IntfName	Nombre de la interfaz

  - Salida:
    - Objeto JSON conteniendo un string: True or false

A.1.2. Information Retriever

Las funciones del Information Retriever son las siguientes:

- **GetFullTopology**: Devuelve la topología de una capa y dominio concretos.

- Parámetros de entrada:
  - http://URL/TopologyModule/GetFullTopology
  - JSON String (<Key, Value>) conteniendo los siguientes parámetros:

Field	***Description***
Layer (optional field)	Capa de la topología a devolver.
DomainID	Dominio de la topología a devolver.

- Salida:
  - Objeto JSON con la estructura:

{"nodeID":"ADVA_NODE_4","address":["172.16.1.40"],"isPhysical":true,"intfList":[{"name":"ADVA_NODE_4_Shelf1-13","address":["41.41.41.1"],"layering":["transport"],"isPhysical":true,"intfUp":true},{"name":"ADVA_NODE_4_Shelf1-10","address":["40.40.40.1"],"layering":["transport"],"isPhysical":true,"intfUp":true},{"name":"ADVA_NODE_4_Unnumbered1","address":["1"],"layering":["transport"],"isPhysical":true,"intfUp":true},{"name":"ADVA_NODE_4_Unnumbered2","address":["2"],"layering":["transport"],"isPhysical":true,"intfUp":true}],"domain":1,"layer":"transport"}{"nodeID":"ADVA_NODE_2","address":["172.16.1.36"],"isPhysical":true,"intfList":[{"name":"ADVA_NODE_2_Unnumbered1","address":["1"],"layering":["transport"],"isPhysical":true,"intfUp":true},{"name":"ADVA_NODE_2_Unnumbered2","address":["2"],"layering":["transport"],"isPhysical":true,"intfUp":true},{"name":"ADVA_NODE_2_Unnumbered3","address":["3"],"layering":["transport"],"isPhysical":true,"intfUp":true}],"domain":1,"layer":"transport"}{"nodeID":"ADVA_NODE_3","address":["172.16.1.38"],"isPhysical":true,"intfList":[{"name":"ADVA_NODE_3_Shelf1-7","address":["31.31.31.1"],"layering":["transport"],"isPhysical":true,"intfUp":true},{"name":"ADVA_NODE_3_Shelf1-14","address":["30.30.30.1"],"layering":["transport"],"isPhysical":true,"intfUp":true},{"name":"ADVA_NODE_3_Unnumbered1","address":["1"],"layering":["transport"],"isPhysical":true,"intfUp":true},{"name":"ADVA_NODE_3_Unnumbered2","address":["2"],"layering":["transport"],"isPhysical":true,"intfUp":true}],"domain":1,"layer":"transport"}{"nodeID":"ADVA_NODE_1","address":["172.16.1.34"],"isPhysical":true,"intfList":[{"name":"ADVA_NODE_1_Shelf1-11","address":["21.21.21.1"],"layering":["transport"],"isPhysical":true,"intfUp":true},{"name":"ADVA_NODE_1_Shelf1-9","address":["20.20.20.1"],"layering":["transport"],"isPhysical":true,"intfUp":true},{"name":"ADVA_NODE_1_Unnumbered1","address":["1"],"layering":["transport"],"isPhysical":true,"intfUp":true}],"domain":1,"layer":"transport"}{"linkID":"ADVA_NODE_4_ADVA_NODE_3","isDirectional":true,"source":{"node":"ADVA_NODE_4","intf":"ADVA_NODE_4_Unnumbered1"},"dest":{"node":"ADVA_NODE_3","intf":"ADVA_NODE_3_Unnumbered1"},"type":"intradomain","teMetric":0.0}{"linkID":"ADVA_NODE_4_ADVA_NODE_2","isDirectional":true,"source":{"node":"ADVA_NODE_4","intf":"ADVA_NODE_4_Unnumbered2"},"dest":{"node":"ADVA_NODE_2","intf":"ADVA_NODE_2_Unnumbered1"},"type":"intradomain","teMetric":0.0}{"linkID":"ADVA_NODE_3_ADVA_NODE_2","isDirectional":true,"source":{"node":"ADVA_NODE_3","intf":"ADVA_NODE_3_Unnumbered2"},"dest":{"node":"ADVA_NODE_2","intf":"ADVA_NODE_2_Unnumbered2"},"type":"intradomain","teMetric":0.0}{"linkID":"ADVA_NODE_2_ADVA_NODE_1","isDirectional":true,"source":{"node":"ADVA_NODE_2","intf":"ADVA_NODE_2_Unnumbered3"},"dest":{"node":"ADVA_NODE_1","intf":"ADVA_NODE_1_Unnumbered1"},"type":"intradomain","teMetric":0.0}

- **GetOppositeNode**: devuelve el nodo opuesto al nodo dado, es decir, aquel que pertenece a la otra capa del nodo introducido.
  - Parámetros de entrada:
    - http://URL/TopologyModule/GetOppositeNode
    - JSON String (<Key, Value>) conteniendo los siguientes parámetros:

Field	***Description***
intfName	Nombre de la interfaz.
Address	Dirección de la interfaz.
Layer	Capa a la que pertenece la interfaz.

IsPhysical	Indica si la interfaz es fisica o no.

- Salida:
  - Objeto JSON con la estructura:

{"node":"ADVA_NODE_4","intf":"ADVA_NODE_4_Shelf1-13"}

- **GetOppositeInterface**: Retrieves the interface in the other layer to which the interface is connected.
  - Parámetros de entrada:
    - http://URL/TopologyModule/ GetOppositeInterface
    - JSON String (<Key, Value>) conteniendo los siguientes parámetros:

Field	***Description***
intfName	Nombre de la interfaz.
Address	Dirección de la interfaz.
Layer	Capa a la que pertenece la interfaz.
IsPhysical	Indica si la interfaz es fisica o no.

  - Salida:
    - Objeto JSON con el nombre de la interfaz opuesta.

- **GetNeighboursNodesOf**: devuelve los nodos vecinos de un nodo pertenecientes a su misma capa.
  - Parámetros de entrada:
    - http://URL/TopologyModule/ GetNeighboursNodesOf
    - JSON String (<Key, Value>) conteniendo los siguientes parámetros:

Field	***Description***
NodeName	Nombre del nodo.

  - Salida:
    - Objeto JSON con la estructura:
      - NodeID.
      - Address.
      - IsPhysical.
      - IntfList
        - Name
        - Address
        - Layering
        - isPhysica
        - intfUp
      - domain
      - location (optional)
      - parentRouter (optional)
      - layer: 'transport' or 'IP'.
    - Ejemplo:

{"nodeID":"ADVA_NODE_4","address":["172.16.1.40"],"isPhysical":true,"intfList":[{"name":"ADVA_NODE_4_Shelf1-13","address":["41.41.41.1"],"layering":["transport"],"isPhysical":true,"intfUp":true},{"name":"ADVA_NODE_4_Shelf1-

10","address":["40.40.40.1"],"layering":["transport"],"isPhysical":true,"intfUp":true},{"name":"ADVA_NODE_4_Unnumbered1","address":["1"],"layering":["transport"],"isPhysical":true,"intfUp":true},{"name":"ADVA_NODE_4_Unnumbered2","address":["2"],"layering":["transport"],"isPhysical":true,"intfUp":true}],"domain":1,"layer":"transport"}{"nodeID":"ADVA_NODE_3","address":["172.16.1.38"],"isPhysical":true,"intfList":[{"name":"ADVA_NODE_3_Shelf1-7","address":["31.31.31.1"],"layering":["transport"],"isPhysical":true,"intfUp":true},{"name":"ADVA_NODE_3_Shelf1-14","address":["30.30.30.1"],"layering":["transport"],"isPhysical":true,"intfUp":true},{"name":"ADVA_NODE_3_Unnumbered1","address":["1"],"layering":["transport"],"isPhysical":true,"intfUp":true},{"name":"ADVA_NODE_3_Unnumbered2","address":["2"],"layering":["transport"],"isPhysical":true,"intfUp":true}],"domain":1,"layer":"transport"}{"nodeID":"ADVA_NODE_1","address":["172.16.1.34"],"isPhysical":true,"intfList":[{"name":"ADVA_NODE_1_Shelf1-11","address":["21.21.21.1"],"layering":["transport"],"isPhysical":true,"intfUp":true},{"name":"ADVA_NODE_1_Shelf1-9","address":["20.20.20.1"],"layering":["transport"],"isPhysical":true,"intfUp":true},{"name":"ADVA_NODE_1_Unnumbered1","address":["1"],"layering":["transport"],"isPhysical":true,"intfUp":true}],"domain":1,"layer":"transport"}

ANEXO B. Publicaciones

1. M. Chamania, O. Gonzalez de Dios, V. López, M. Cuaresma, A. Jukan, M. Drogon, X. Masip-Bruin, Marcelo Yannuzzi, Coordinated Computation of Multi-layer Paths via Inter-layer PCE Communication: Standards, Interoperability and Deployment, in the Proc. of the Workshop on Telecommunications: From Research to Standards at ICC, June 2012.

2. O. Gonzalez de Dios, M. Cuaresma, S. Martinez, F. Munoz, V. López, J.P. Fernandez-Palacios, Functional validation of the cooperation between Virtual Network Topology Manager (VNTM) and Path Computation Element (PCE), in International Conference on IP + Optical Network (iPOP), Jun, 2012.

3. O. Gonzalez de Dios, V. López, M. Cuaresma, F. Muñoz, M. Chamania and A. Jukan: Coordinated Computation and Setup of Multi-layer Paths via Inter-layer PCE Communication: Standards, Interoperability and Deployment, accepted for its publication in IEEE Communications Magazine.

4. M. Cuaresma, F. Muñoz, S. Martinez, A. Mayoral, O. Gonzalez de Dios, V. López, J.P. Fernández-Palacios, Experimental Demonstration of H-PCE with BGP-LS in elastic optical networks, in European Conference on Optical Communication (ECOC), We.4.E.3, Sep 2013

REFERENCIAS

[1] D. King and A. Farrel, "A PCE-based Architecture for Application-based Network Operations". Disponible en: http://tools.ietf.org/html/draft-farrkingel-pce-abno-architecture-05.

[2] A. Farrel, J.P. Vasseur, and J. Ash, "A path computation element (PCE)-based architecture," IETF RFC 4655, pp. 1–40, Aug. 2006.

[3] JL. Le Roux and JP. Vasseur, "Path Computation Element (PCE) Communication Protocol (PCEP)", IETF RFC 5440, pp. 1-87, March, 2009. Disponible en: http://tools.ietf.org/html/rfc5440.

[4] D. Álvarez, V. López, J.L. Añamuro, J.E. López de Vergara, O. González de Dios and J. Aracil: "Utilization of Temporary Reservation of Path Computed Resources for Multi-Domain PCE Protocols in WDM Networks", in the proceeding of Network of the Future conference, Nov, 2011.RFC 4655.

[5] O. González de Dios, R. Casellas, C. Margaria, Y. Lee and F. Zhang, "PCEP Extensions for Temporary Reservation of Computed Path Resources and Support for Limited Context State in PCE", IETF draft-gonzalezdedios-pce-reservation-state-01, March 2012.

[6] J. Seedorf and E.Burguer, "Application-Layer Traffic Optimization (ALTO) Problem Statement", IETF RFC 5693. October 2009. Disponible en: http://tools.ietf.org/html/rfc5693.

[7] Nick McKeown, Tom Anderson, Hare Balakrishnan, Guru Parulkar, Larry Peterson Jennifer Rexford, Scott Shenker and Jonathan Turner, "OpenFlow: Enabling Innovation in Campus Networks", March 2008. Disponible en: http://www.openflow.org/documents/openflow-wp-latest.pdf.

[8] A. Atlas, J. Halpern, S. Hares, D. Ward, T. Nadeau, "An Architecture for the Interface to the Routing System", July 15, 2013, IETF draft-atlas-i2rs-architecture.

[9] B. Niven-Jenkins, F. Le Faucheur and N. Bitar, "Content Distribution Network Interconnection (CDNI) Problem Statement", IETF RFC 6707, September 2012. Disponible en: http://tools.ietf.org/html/rfc6707.

[10] Y. Lee, JL. Le Roux, D. King, E. Oki, "Path computation element communication protocol (PCEP) requirements and protocol extensions in support of global concurrent optimization," IETF RFC 5557, July 2009.

[11] E. Crabbe, J. Medved, I. Minei and R. Varga, "PCEP Extensions for Stateful PCE". June 30, 2013. Disponible en: http://tools.ietf.org/html/draft-ietf-pce-stateful-pce-05.

[12] K. Shiomoto, D. Papadimitriou, JL. Le Roux, M. Vigoureux and D. Brungrad, "Requirements for GMPLS-Based Multi-Region and Multi-Layer Networks (MRN/MLN)". IETF RFC 5212, July 2008. Disponible en: http://tools.ietf.org/html/rfc5212.

[13] E. Oki, T. Takeda, JL. Le Roux, A. Farrel, "Framework for PCE-Based Inter-Layer MPLS and GMPLS Traffic Engineering", Septembre 2009. Disponible en: http://tools.ietf.org/html/rfc5623

[14] O. Gonzalez de Dios, V. López, M. Cuaresma, F. Muñoz, M. Chamania and A. Jukan, "Coordinated Computation and Setup of Multi-layer Paths via Inter-layer PCE Communication: Standards, Interoperability and Deployment", IEEE Communications Magazine.

[15] V. López, B. Huiszoon, J. Fernández-Palacios, O. González de Dios and J. Aracil, "Path Computation Element in Telecom Networks: Recent Developments and Standardization Activities", ONDM, Feb 2010.

[16]J. Ash and J.L. Le Roux, "Path computation element (PCE) communication protocol generic requirements", RFC 4657, pp. 1-21, September 2006. Disponible en: http://tools.ietf.org/html/rfc4657.

[17]J. Moy, "OSPF Version 2", RFC 2328, April 1998. Disponible en: http://www.ietf.org/rfc/rfc2328.

[18]O. González de Dios, J. Jimenez Chico, F. Muñoz del Nuevo, "Benefits of limited context awareness in stateless PCE", OFC Conference, March 6, 2011.

[19]A. Mokhtar y M. Azizoglu, "Adaptive wavelength routing in all-optical networks", IEEE/ACM transactions on Networking, vol. 6, no.2 pp. 197 - 201, April 1998.

[20]D. Awduche, L. Berger, D. Gan, T.Li, V. Srinivasan, and G. Swallow , "RSVP-TE: Extensions to RSVP for LSP Tunnels.", RFC 3209, December 2001.

[21]K. Kompella and Y. Rekhter, "Signalling Unnumbered Links in Resource ReSerVation Protocol - Traffic Engineering (RSVP-TE)", RFC 3477, January 2003.

[22]T. Otani and D. Li, "Generalized Labels for Lambda-Switch-Capable (LSC) - Label Switching Routers", RFC 6205, March 2011.

[23]H. Gredler, J. Medved, S. Previdi, A. Farrel and S. Ray, "North-Bound Distribution of Link-State and TE Information using BGP", May 21, 2013. Disponible en: http://tools.ietf.org/html/draft-ietf-idr-ls-distribution-03.

[24]Y. Rekhter, T. Li and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006. Disponible en: http://tools.ietf.org/html/rfc4271

[25]"Transmission Control Protocol", RFC 793, September 1981. Disponible en: http://www.ietf.org/rfc/rfc793.txt

[26]T. Bates, R. Chandra, D. Katz and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, January 2007.

[27]G. Bernstein, Y. Lee, D. Li and W. Imajuku, "General Network Element Constraint Encoding for GMPLS Controlled Networks", May 2013, draft-ietf-ccamp-general-constraint-encode-11.

[28]M. Cuaresma, F. Muñoz, S. Martínez, A. Mayoral, O. Gonzalez, V. López and J.Fernández, "Experimental Demonstration of H-PCE with BPG-LS in elastic optical networks", ECOC, Sep 2013.

[29]D. King and A. Farrel, "The Application of the Path Computation Element Architecture to the Determination of a Sequence of Domains in MPLS and GMPLS", RFC 6805, Nov 2012.